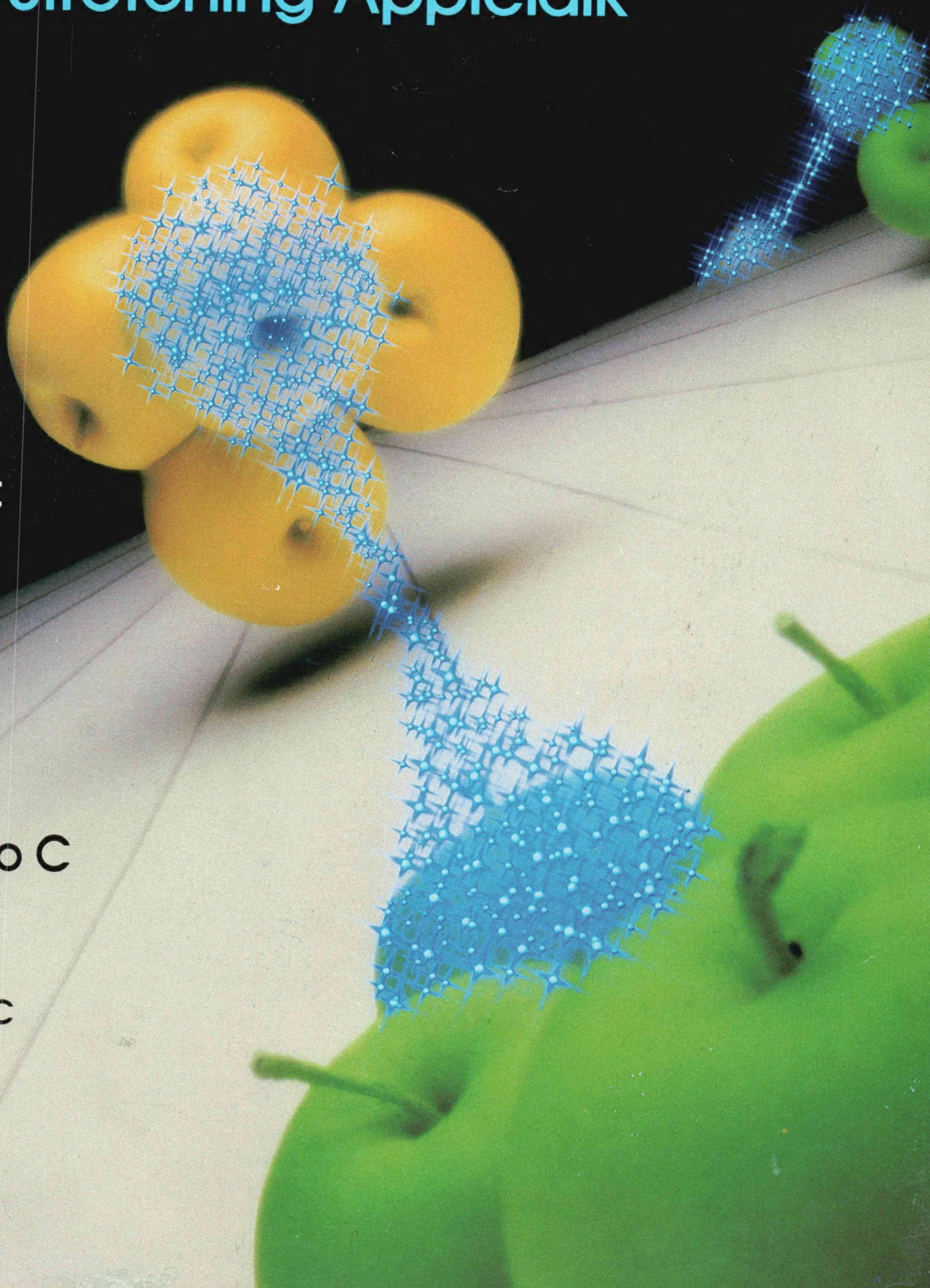
#132 OCTOBER 1987

2.95 (3.95 CANADA)

Dr. Dobb's Journal of Software Tools

FOR THE PROFESSIONAL PROGRAMMER

Stretching AppleTalk



Focus on Forth:

Unifying Dialects
Faster Forth
A New Column

Quick C & Turbo C

Languages:

Forth, Pascal, BASIC, C

10
0 38351 16562 8

DR DOBB'S JOURNAL
132 OCTOBER 1987

bo C: NEW!
erful optimizing
ler ever

Sieve benchmark

	Turbo C	Microsoft® C
Compile time	*2.4*	13.51
Compile and link time	*4.1*	18.13
Execution time	*3.95*	5.93
Object code size	*239*	249
Execution size	*5748*	7136
Price	*$99.95*	$450.00

Benchmark run on an IBM PS/2 Model 60 using Turbo C version 1.0 and the Turbo Linker version 1.0; Microsoft C version 4.0 and the MS overlay linker version 3.51.

Technical Specifications

- ☑ Compiler: One-pass optimizing compiler generating linkable object modules. Included is Borland's high-performance Turbo Linker.™ The object module is compatible with the PC-DOS linker. Supports tiny, small, compact, medium, large, and huge memory model libraries. Can mix models with near and far pointers. Includes floating point emulator (utilizes 8087/80287 if installed).
- ☑ Interactive Editor: The system includes a powerful, interactive full-screen text editor. If the compiler detects an error, the editor automatically positions the cursor appropriately in the source code.
- ☑ Development Environment: A powerful "Make" is included so that managing Turbo C program development is highly efficient. Also includes pull-down menus and windows.
- ☑ Links with relocatable object modules created using Borland's Turbo Prolog into a single program.
- ☑ Inline assembly code.
- ☑ Loop optimizations.
- ☑ Register variables.
- ☑ ANSI C compatible.
- ☑ Start-up routine source code included.
- ☑ Both command line and integrated environment versions included.
- ☑ License to the source code for Run-time Library available.

Join more than 100,000 Turbo C enthusiasts. Get your copy of Turbo C today!

Minimum system requirements: All products run on IBM PC, XT, AT, PS/2, portable and true compatibles. PC-DOS (MS-DOS) 2.0 or later. 384K RAM minimum. Basic Telecom and Editor Toolboxes require 640K.

Borland International
4585 Scotts Valley Drive, Scotts Valley, CA 95066
Telephone: (408) 438-8400 Telex: 172373

BI-1131 A

Why more than 600,000 programmers worldwide are using Turbo Pascal today

The irresistible force behind Turbo Pascal's worldwide success is Borland's advanced technology. We created a compiler so fast, that Turbo Pascal® is now the worldwide standard. And there are more tools for Turbo Pascal than for any other development environment in the world.

You'll get everything you need from Turbo Pascal and its 5 Toolboxes

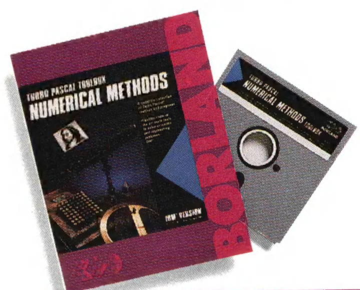
Turbo Pascal and Family are all you'll ever need to perfect programming in Pascal.

If you've never programmed in Pascal, you'll probably want to start with Turbo Pascal Tutor® 2.0, and as your expertise quickly grows, add Toolboxes like our

- Database Toolbox®
- Editor Toolbox®
- Graphix Toolbox®
- GameWorks®
  and our newest,
- Numerical Methods Toolbox™

And because Turbo Pascal is the established worldwide standard, 3rd party, independent non-Borland developers also offer an incredible array of programs for Turbo Pascal. ***Only $99.95!***

> "Borland International's Turbo Pascal took the programming world by storm. A great compiler combined with a good editor at an astounding price, the package quickly came to be called, simply, Turbo—and has sold more than 500,000 copies.
>
> *Stephen Randy Davis, PC Magazine*
>
> Language deal of the century. *PC Magazine*"



For Scientists and Engineers: Turbo Pascal Numerical Methods Toolbox

The Numerical Methods Toolbox is a complete collection of Turbo Pascal routines and programs. Add it to your development system and you have the most comprehensive and powerful numerical analysis capabilities—at your fingertips!

The Numerical Methods Toolbox is a state-of-the-art mathematical toolbox with these ten powerful features:

- ☑ Zeros of a function
- ☑ Interpolation
- ☑ Differentiation
- ☑ Integration
- ☑ Matrix Inversion
- ☑ Matrix Eigenvalues
- ☑ Differential Equations
- ☑ Least Squares
- ☑ Fourier Transforms
- ☑ Graphics

Each module comes with procedures that can be easily adapted to your own program. The Toolbox also comes complete with source code. So you have total control of your application.

Only $99.95!

Turbo C, Turbo Basic, Turbo Pascal and Turbo Prolog: technical excellence

"Borland International's Turbo Pascal, Turbo Basic and Turbo Prolog automatically identify themselves, by virtue of their 'Turbo' forenames, as superior language products with a common programming environment. The appellation also means to many PC users a 'must have' language. To us Turbo C looks like a coup for Borland.

Garry Ray, PC Week"

Turbo Prolog: The Natural Language of Artificial Intelligence

Whether you're a first-time programmer or an experienced one, Turbo Prolog's natural implementation of Artificial Intelligence soon shows you how to build expert systems, natural language interfaces, customized knowledge bases and smart information management systems.

Turbo Prolog and Turbo C work hand-in-hand

Turbo Prolog® interfaces perfectly with Turbo C® because they're both designed to work with each other.

The Turbo Prolog/Turbo C combination means that you can now build powerful commercial applications using two of the most powerful languages available.

Turbo Prolog's development system includes:

- ☑ A complete Prolog compiler that is a variation of the Clocksin and Mellish Edinburgh standard Prolog.
- ☑ A full-screen interactive editor.
- ☑ Support for both graphic and text windows.
- ☑ All the tools that let you build your own expert systems and **AI** applications with unprecedented ease.

All Borland products are trademarks or registered trademarks of Borland International, Inc., or Borland/Analytica, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.
Copyright 1987 Borland International BI-1131 A

"An affordable, fast, and easy-to-use language that will delight the newcomer . . . You experienced Prolog hackers will likewise be delighted, if not astonished, by the features and performance of the Turbo Prolog development environment.

Turbo Prolog offers generally the fastest and most approachable implementation of that language."

Darryl Rubin, AI Expert

How Turbo Prolog's new Toolbox adds 80 powerful tools and 8000 lines of source code

In keeping with Borland tradition, we've quickly added the new Turbo Prolog Toolbox™ to Turbo Prolog.

With 80 tools and 8000 lines of source code that can easily be incorporated into your own programs—and 40 sample programs that show you how to put these AI tools to work—the Turbo Prolog Toolbox is a highly intelligent, high-performance addition.

Only $99.95!

Turbo Prolog Toolbox features include:

- ☑ Business graphics generation: boxes, circles, ellipses, bar charts, pie charts, scaled graphics
- ☑ Complete communications package: supports XMODEM protocol
- ☑ File transfers from Reflex,® dBASE III,® 1-2-3,® Symphony®
- ☑ A unique parser generator: construct your own compiler or query language
- ☑ Sophisticated user-interface design tools
- ☑ Contains 40 example programs
- ☑ Easy-to-use screen editor: design your screen layout and I/O
- ☑ Calculated fields definition
- ☑ Over 8,000 lines of source code you can incorporate into your own programs

Tur[bo C:] The most pow[erful] compi[ler]

Our new Turbo C generates fast, tight, production-quality code at compilation speeds of more than 13,000 lines a minute!

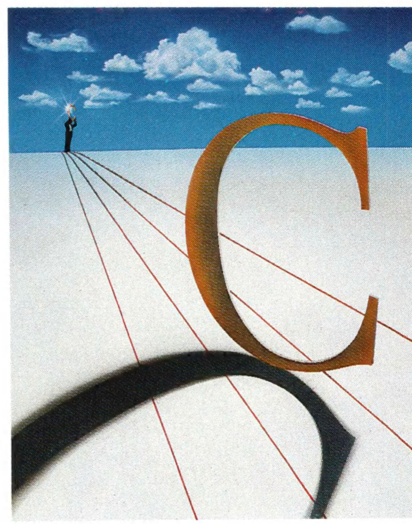
It's the full-featured optimizing compiler everyone has been waiting for.

Switching to Turbo C, or starting with Turbo C, you win both ways

If you're already programming in C, switching to Turbo C will make you feel like you're riding a rocket instead of pedaling a bike.

If you're never programmed in C, starting with Turbo C gives you an instant edge. It's easy to learn, easy to use, and the most efficient C compiler at any price.

Only $99.95!



"Turbo C does look like What We've All Been Waiting For: a full-featured compiler that produces excellent code in an unbelievable hurry . . . moves into a class all its own among full-featured C compilers . . . Turbo C is indeed for the serious developer . . . One heck of a buy—at any price."

Michael Abrash, Programmer's Journal

Turbo Basic introduces its powerful new Telecom, Editor and Database Toolboxes

NEW!

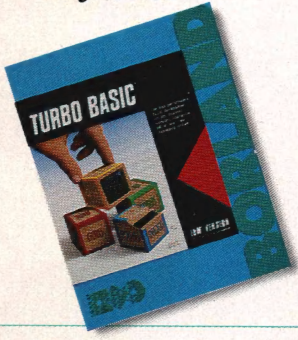
Turbo Basic® is the breakthrough you've been waiting for. The same power we brought to Pascal with Turbo Pascal has now been applied to BASIC with Turbo Basic.

Compatible with BASICA, Turbo Basic is the high-performance, high-speed BASIC you'd expect from Borland.

Basically, Turbo Basic is all you need

It's a complete development environment which includes an incredibly fast compiler, an interactive editor and a trace debugging system. It outperforms all its rivals, and because it's compatible with BASICA, you probably already know how to use it.

Includes a free MicroCalc™ spreadsheet complete with source code. ***Only $99.95!***



A technical look at Turbo Basic

- ☑ Full recursion supported
- ☑ Standard IEEE floating-point format
- ☑ Floating-point support, with full 8087 (math co-processor) integration. Software emulation if no 8087 present
- ☑ Program size limited only by available memory (no 64K limitation)
- ☑ VGA, CGA, and EGA support
- ☑ Access to local, static, and global variables
- ☑ Full integration of the compiler, editor, and executable program, with separate windows for editing, messages, tracing, and execution
- ☑ Compile, run-time, and I/O errors place you in the source code where error occurred
- ☑ New long integer (32-bit) data type
- ☑ Full 80-bit precision
- ☑ Pull-down menus
- ☑ Full window management

> "Borland has created the most powerful version of BASIC ever.
> *Ethan Winer, PC Magazine*"

NEW!

Telecom Toolbox is a complete communications package which takes advantage of the built-in communications capabilities of BASIC—use as is or modify.

- Pull-down menus and windows
- XMODEM support
- VT 100 terminal emulation
- Captures text to disk or printer
- PhoneBook file
- 300, 1200, 2400 baud support
- Supports script files
- Fast screen I/O
- Supports most of XTalk's command set
- Manual dial and redial options

Use Telecom Toolbox to embed communications capabilities into your own programs and/or build your own communications package. Source code included for all Toolbox code and sample programs. ***Only $99.95!***

For the dealer nearest you or to order by phone call

(800) 255-8008

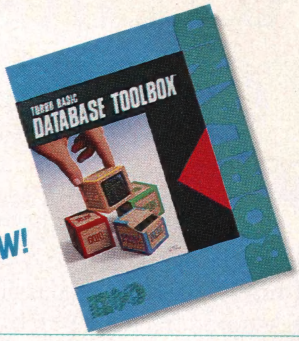
in CA (800) 742-1133 in Canada (800) 237-1136

BORLAND
INTERNATIONAL

SUMMER BREAK SPECIAL ! Buy Turbo Basic and get a FREE product. See your dealer for details!

CIRCLE 161 ON READER SERVICE CARD

NEW!



Database Toolbox means that you don't have to reinvent the wheel each time you write new Turbo Basic database programs.

- ☑ "Trainer" shows you how B+ trees work. (Simply key in sample records and you'll see your index being built.)
- ☑ Turbo Access instantly locates, inserts or deletes records in a database—using B+ trees.
- ☑ Turbo Sort sorts data on single items or on multiple keys and features virtual memory management for sorting large data files.

Source code included.
Only $99.95!

NEW!

Editor Toolbox is all you need to build your own text editor or word processor. Includes source code for two sample editors.

First Editor is a complete editor ready to include in your programs, complete with windows, block commands and memory-mapped screen routines.

MicroStar™ is a full-blown text editor with a complete pull-down menu user interface, and gives you

- Wordwrap
- Undo last change
- Auto-Indent
- Find and Find/Replace with options
- Set left/right margins
- Block mark, move and copy
- Tab, insert, overstrike modes, line center etc.

Includes source code.
Only $99.95!

BI-1131A

Now, just $129 and supports Microsoft C 5.0 and QuickC

Blaise puts the Accent on C with C TOOLS PLUS/5.0™

Enhance your Microsoft C programming environment with C TOOLS PLUS/5.0™—a new, quintessential library of C functions. C TOOLS PLUS/5.0 from Blaise Computing Inc. puts a prime accent on quickly building professional applications using the full power of Microsoft C Version 5.0 and QuickC. Now you can concentrate on program creativity by having full control over DOS, menus, interrupt service routines, memory resident programs, printer and keyboard control, and more!

C TOOLS PLUS/5.0 prebuilt libraries are ready to use with either QuickC or the Microsoft C Version 5.0 command line environment. Complete documented source code is included so that you can study and adapt it to your specific needs. Blaise Computing's attention to detail, like the use of full function prototyping, cleanly organized header files, and a comprehensive, fully-indexed manual, makes C TOOLS PLUS/5.0 the choice for experienced developers as well as newcomers to C.

Continuous refinement of Blaise Computing's library products has produced a collection of tools that are unsurpassed for reliability, functionality and ease of use. Built upon the widely acclaimed C TOOLS PLUS, C TOOLS PLUS/5.0 includes such highly-developed features as:

◆**WINDOWS**
—Stackable, removable.
—Optional borders, cursor memory.
—Accept user input, formatted output.
—"printf" window-oriented output. **NEW!**

◆**INTERRUPT SERVICE ROUTINES**
—Capture DOS critical errors and keystrokes.
—Install hardware interrupt handlers.

◆**RESIDENT SOFTWARE SUPPORT**
—Install, detect and remove memory resident programs.

◆**MENUS**
—Horizontal and pulldown. **NEW!**
—Lotus-style support. **NEW!**

◆**INTERVENTION CODE**
—Schedule C functions at specified times, intervals or with a "hot key." **NEW!**
—Take full advantage of DOS, even from memory resident programs. **NEW!**

◆**FAST DIRECT VIDEO ACCESS**
—All monitors, even EGA 43-line mode.

◆**PRINTER CONTROL**
—Access BIOS print functions. **NEW!**
—Control the DOS PRINT utility. **NEW!**

◆**UTILITIES AND MACROS**
—Take advantage of DOS file structure.
—Manipulate data types, far & near pointers. **NEW!**
—Access any memory areas with fast "peek" and "poke" macros. **NEW!**

C TOOLS PLUS/5.0 supports the Microsoft C Version 5.0 and QuickC compilers, requires DOS 2.00 or later and is just **$129.00.**

C ASYNCH MANAGER™ Version 2.0 IMPROVED!

C ASYNCH MANAGER is a library of functions designed to help you incorporate asynchronous communication capabilities into your application programs. Version 2.0 has been rewritten especially for Microsoft C Version 5.0 and Borland's Turbo C. Simultaneous buffered input and output to both COM ports at speeds up to 9600 baud, XON/XOFF protocol, modem control and XMODEM file transfer are among the many features supported and is priced at just **$175.00.**

Blaise computing Inc. has a full line of support products for both Pascal and C. Call today for your free information packet.

BLAISE COMPUTING INC.
2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

CIRCLE 159 ON READER SERVICE CARD

THE BLAISE MENU

VIEW MANAGER *$275.00*
General screen control; paint screens; block mode data entry or field-by-field control with instant screen access. For C or MS-Pascal.

Turbo C TOOLS *$129.00*
Windows; ISRs; intervention code; screen handling and EGA 43-line text mode support; direct screen access; DOS file handling and more. For Turbo C.

Turbo POWER SCREEN
COMING SOON! General screen management; paint screens; block mode data entry or field-by-field control with instant screen access. For Turbo Pascal.

Turbo POWER TOOLS PLUS *$99.95*
Screen and window management including EGA support; DOS memory control; ISRs; scheduled intervention code; and much more. For Turbo Pascal.

Turbo ASYNCH PLUS *$99.95*
Interrupt driven support for the COM ports. I/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem and XMODEM control. For Turbo Pascal.

PASCAL TOOLS/TOOLS 2 *$175.00*
Expanded string and screen handling; graphics routines; memory management; general program control; DOS file support and more. For MS-Pascal.

ASYNCH MANAGER *$175.00*
Full featured interrupt driven support for the COM ports. I/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem control and XMODEM. For MS-Pascal.

KeyPlayer *$49.95*
"Super-batch" program. Create batch files which can invoke programs and provide input to them; run any program unattended; create demonstration programs; analyze keyboard usage.

EXEC *$95.00*
NEW VERSION! Program chaining executive. Chain one program from another in different languages; specify common data areas; less than 2K of overhead.

RUNOFF *$49.95*
Text formatter for all programmers; flexible printer control; user-defined variables; index generation; general macro facility. Crafted in Turbo Pascal.

LIGHT TOOLS *$99.95*
Windows; ISRs; EGA 43-line text mode; direct screen access; DOS file handling and more. For the Datalight C compiler.

TO ORDER CALL TOLL FREE 800-333-8087

TELEX NUMBER-338139

Yes! Send me the prime accent!
Enclosed is $________ for ________ copies of ________.
☐ Please send me more information on your products.
CA residents add Sales Tax. Domestic orders add $4.00 for UPS shipping, $10.00 for Federal Express standard air.
Name: ________ Phone: (____) ____-____
Address: ________ State: ____ Zip: ____
City: ________
VISA or MC#: ________ Exp. Date: ____

Microsoft is a registered trademark of Microsoft Corporation. QuickC is a trademark of Microsoft Corporation. Turbo C is a registered trademark of Borland International.

OCTOBER 1987

CONTENTS

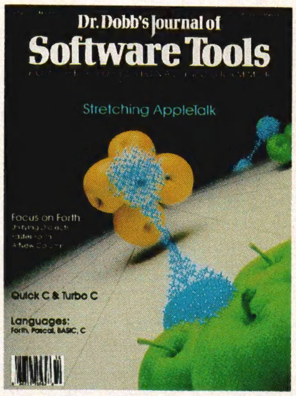
VOLUME 12, ISSUE 10

ARTICLES

COLUMNS

FORUM

PROGRAMMER'S SERVICES



About the Cover

"Apples in space" is just our way of saying that physical distance, be it a hundred feet or a few thousand miles, is no longer an obstacle to linking AppleTalk nodes.

This Issue

Our annual Forth issue introduces a new bimonthly Forth column as well as two ways to increase Forth speed: the first increases execution speed on 68000 machines; the second trims development time for PC Forths. Finally, our cover story looks at how two gentlemen from Dartmouth have expanded AppleTalk into a "nonlocal" area network.

Next Issue

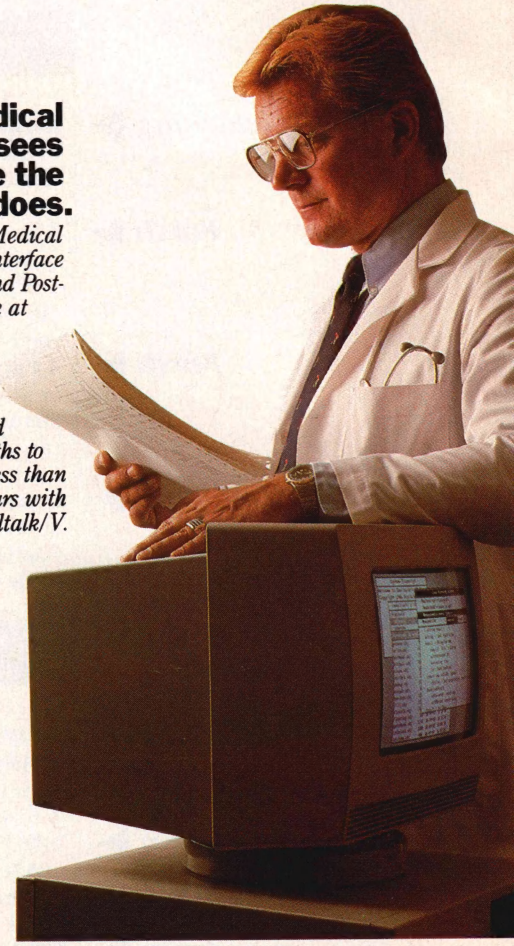
November's *DDJ* features a graphics theme, and we're not talking about just another pretty face here, folks. Programming PC graphics is the focus, and we'll approach it from several different language directions.

This ad is for people who don't know where to find Smalltalk. Or why.

Today, the single most important emerging software technology is OOPS, object-oriented programming. It's destined to dramatically change the way you use your personal computer. You'll find it doing things you never expected. And by people you never suspected.

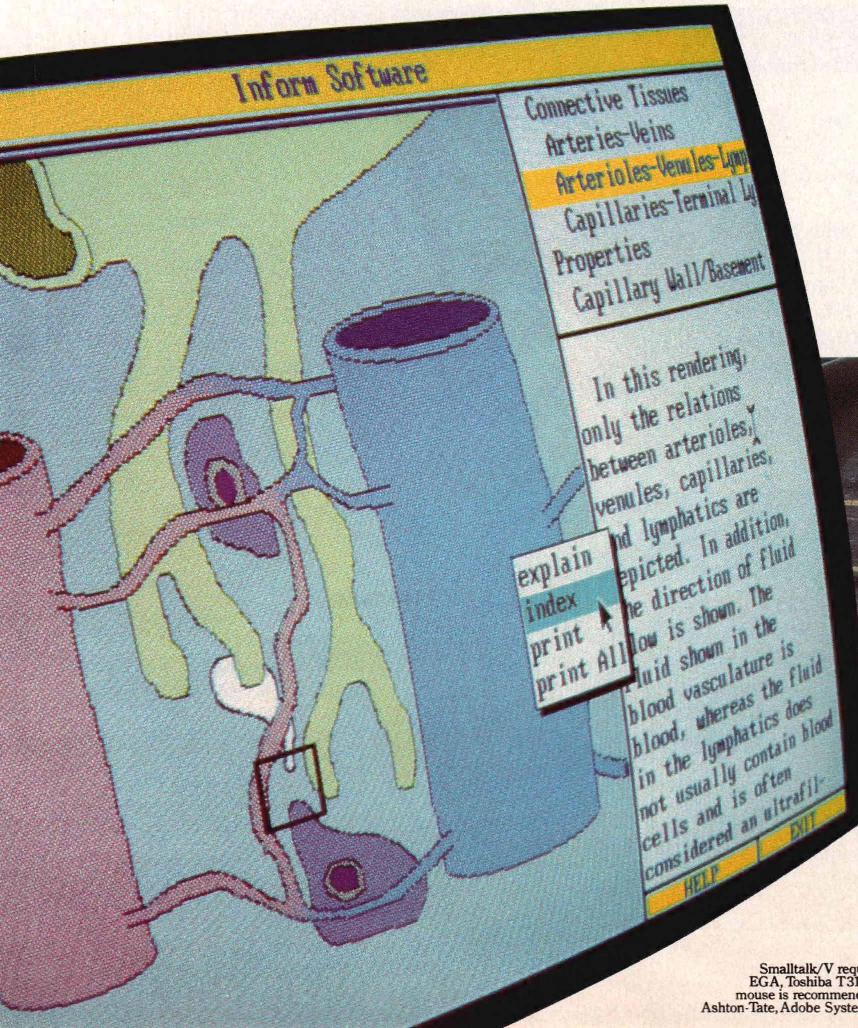
At the UCLA Medical Center, it sees patients before the doctor does.

Mike McCoy, M.D., at the UCLA Medical Center, found that he could easily interface Smalltalk/V with dBASEIII and PostScript. His application, now in use at the Clinic, turns a functional status questionnaire on each new patient into a laser printed, advisary analysis for the doctor to review prior to seeing the patient. A program like this would normally take a specialist months to produce. It took Dr. McCoy less than 100 hours with Smalltalk/V.



In an emergency room in Vancouver, it's saving lives through animation.

What if a medical textbook could come to life? What if it could show the effects emergency treatment might have on patients? And do it all through moving pictures? These thoughts led Folkstone Design, Edge Training & Consulting, and Inform Software in Vancouver, B.C., to create the first animated, interactive textbook for emergency room technicians and in-training paramedics. They found Smalltalk/V could easily facilitate a combination of text, color graphics and animation to illustrate various physical processes and the results of medical intervention.



It's working on Florida's freeways.

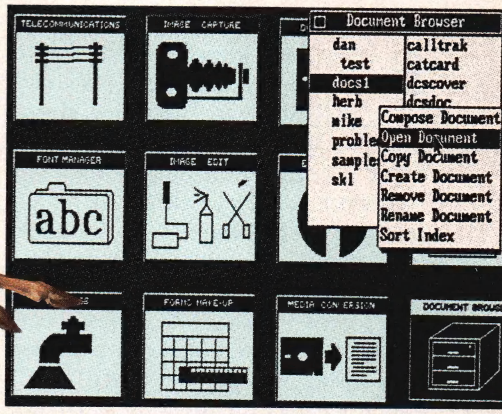
Running on IBM's new PS/2, a Smalltalk/V application developed by Greiner Engineering's Mike Rice, lets highway engineers create highly sophisticated graphic analyses of any proposed reconstruction. So now, instead of having to deal with a gridlock of Federal and State regulations, engineering specifications and endless calculations, an engineer can quickly explore alternative design strategies using a mouse, windows and VGA color graphics.

Smalltalk/V requires DOS and 512K RAM on IBM PC/AT/PS or compatibles and a CGA, EGA, Toshiba T3100, Hercules, or AT&T 6300 graphic controller. A Microsoft or compatible mouse is recommended. Not copy protected. dBASEIII, PostScript and PS/2 are trademarks of Ashton-Tate, Adobe Systems and International Business Machines Corporation respectively.

You can find it in space.

On a project commissioned by NASA, Dr. Christine Mitchell at the Georgia Institute of Technology, chose to use Smalltalk/V as an integral part of a new man-machine interface. The application, written in Smalltalk, continually monitors the commands of the Satellite Network Operator, the state-of-the-network and the overall mission plans.

To NASA, Smalltalk/V means real-time. Real OOPS. Real results.



It's tracking white-tail deer on the Barrier Islands of Georgia.

Dr. Lee Graham, a National Park Service ecologist chose Smalltalk/V to write an application to help manage the white-tail deer population on the Barrier Islands of Georgia. Dr. Graham found that Smalltalk/V, with its visual interface and class structure, is a perfect tool to graphically simulate the complex, ecological interactions of natural systems.

It's making headlines in Arizona.

When Digital Composition Systems sat down to build an electronic typesetting system, they had three major requirements. It had to have the most advanced user interface. It had to be fast. And, it had to be able to turn untrained personnel into high quality typographers. Of all the languages in the world, they chose Smalltalk/V. The result is the Signature Series, recognized and reviewed by The Seybold Report. It's now marketed by Digital Composition Systems and one of the largest digital typesetting firms in the world, Varityper AM International.

What thousands of people have found is OOPS.

Object-Oriented Programming (OOPS) is programming by defining objects, their inter-relationships and their behavior. Objects can represent both real-world entities like people, places, or things. They can also represent useful abstractions such as stacks, sets and rectangles.

OOPS models the way you think and the way things really are. It lets you solve problems by breaking them down into easily handled sub-problems and their inter-relationships. The solutions you come up with can be re-used to solve new problems. Ultimately, OOPS makes programming a simple, logical process of building on the work of others.

Why thousands more are finding their way to Smalltalk/V.

First of all, Smalltalk/V makes OOPS easy.

It's also fast. In fact, it's the fastest OOPS programming available on a PC.

And it's easy to learn. It comes complete with a tutorial that's the best introduction to OOPS available.

Smalltalk/V also has a few other features worth noting. Like a user-extendable, open ended environment. Source code with browser windows for easy access and modification. A huge toolkit of classes and objects for building a variety of applications. A sophisticated source-level debugger. Object-oriented Prolog integrated with the Smalltalk environment. And bit-mapped graphics with bit and form editors, just to name a few.

Then, there's its unbelievable price of only $99.95. (Optional application packs at $49.95 include Communications, EGA/VGA Color and Goodies.)

And it has a 60 day, money-back guarantee.

With all this to offer, it probably won't come as a surprise to you that more people are solving more problems with Smalltalk/V than any other OOPS.

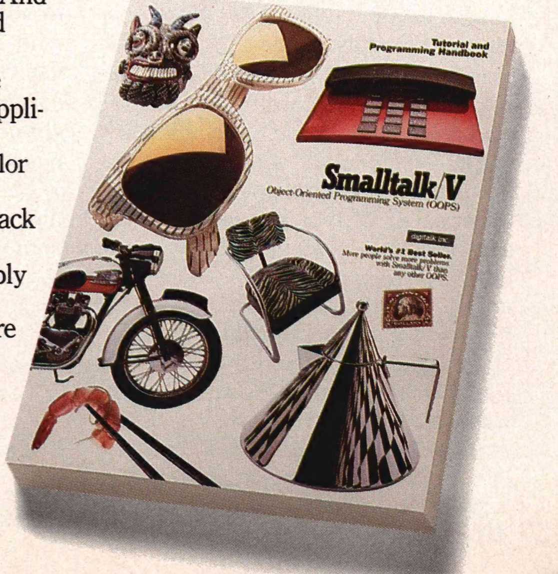
See your nearest dealer today for your own Smalltalk/V. Or, order it direct with MasterCard or Visa at (800) 922-8255.

Or, write to Digitalk, Inc., 9841 Airport Blvd., Los Angeles, CA 90045. Then discover all the great things you can do with your PC and Smalltalk/V.

Smalltalk/V

digitalk inc.

CIRCLE 127 ON READER SERVICE CARD



*Now that you've found us, write us. Tell us some of the great things you're doing with Smalltalk/V. You could be in our next ad.

CIRCLE 127 ON READER SERVICE CARD

EDITORIAL

Lebensraum and RAM

Since the beginning of the micro revolution, *DDJ* readers have been plagued with memory shortages. To be sure, our perception of what constitues "enough" memory has been subject to almost constant revision. In the heyday of the S-100 bus, the introduction of 16K dynamic RAM cards was an exciting development. Now just about everyone in the PC realm is complaining that G40K of RAM simply isn't enough room for all their software Déjà vu.

A few years ago, the folks at Lotus and Intel got together and came up with a way of expanding the amount of memory in a PC beyond the 640K barrier. Before long, Microsoft had gotten into the act, and we had a formal spec: the Lotus/Intel/Microsoft Expanded Memory Specification (EMS). The EMS was admittedly a kludge, but bank-switched RAM was better than no RAM at all.

Given the nature of the market, it was inevitable that some people wouldn't be all that thrilled with some of the limitations of the LIM version. Before long an AST/Quadram/Ashton-Tate coalition produced the Enhanced Expanded Memory Specification, a superset of the original plan.

Mercifully, developers weren't asked to deal with any more versions than these two.

Of course, not everyone was happy with the limits of the EMS and the EEMS. There were still some rough edges in the standards, some confusion in the minds of programmers as to which standard to write for, and that annoying upper limit of eight megabytes. And with the growing realization that IBM's PS/2 line isn't going to instantly make all the older DOS machines obsolete, the importance of hardware and software that maximizes the performance of DOS machines has begun to take on a new importance.

And this is where we have good news to report. Recently the folks at Lotus, Intel, and Microsoft announced a new version of the EMS. Rather than yet another departure in standards, Version 4.0 of the Expanded Memory Standard is a superset of both previous standards and the result of cooperation between the vendors.

Representatives from other companies were present at the press conference announcing the new version. People from AST Research were there, as you might expect, but other companies, such as Ashton-Tate, Borland, and Symantec, were also represented. It was particularly gratifying to see Terry Myers of Quarterdeck Systems announcing products that would take advantage of the new standard at the same time the standard was announced. (At long last, the EMS will support multiple processes.)

Although it may seem at times that we here at *DDJ* can only flame when we hear names such as Lotus, Intel, and Microsoft, that's not really the case. We'd like to take this opportunity to commend the folks at AST, Intel, Lotus, Microsoft, and the other companies involved for cooperating in the new EMS standard. In particular, Microsoft deserves praise for making good on its pledge not to abandon the millions of PC users who aren't going to abandon DOS for OS/2. The older machines will continue to need new, innovative software, and this cooperative EMS effort will ease the burdens of both users and programmers. Indeed, the only question I had after the press conference was "Why only 32 megabytes?"

Tyler Sperry

Tyler Sperry
editor

Dr. Dobb's Journal of Software Tools
FOR THE PROFESSIONAL PROGRAMMER

Editorial
Editor-in-Chief *Michael Swaine*
Editor *Tyler Sperry*
Managing Editor *Vince Leone*
Associate Editor *Ron Copeland*
Assistant Editor *Sara Noah Ruddy*
Technical Editors *Allen Holub*
Richard Relph
Contributing Editors *Namir Shammas*
Ernest R. Tello
Copy Editor *Rhoda Simmons*
Production
Production Manager *Bob Wynne*
Art Director *Michael Hollister*
Assoc. Art Director *Joe Sikoryak*
Technical Illustrator *Frank Pollifrone*
Typesetter *Mary Lopez*
Cover Photographer *Michael Carr*
Circulation
Circulation Director *Maureen Kaminski*
Fulfillment Coordinator *Francesca Martin*
Book Marketing Mgr. *Jane Sharninghouse*
Subscription Supervisor *Kathleen Shay*
Newsstand Sales *Larry Hupman*
Administration
Finance Director *Kate Wheat*
Business Manager *Betty Trickett*
Accounts Payable Supv. *Mayda Lopez-Quintana*
Accts. Receivable Supv. *Laura DiLazzaro*
Advertising Director
Ferris Ferdon (415) 366-3600
Marketing Mgr. *Michael Wiener*
Trafficking Coordinator *Donna Rogers*
Account Managers *see page 129*
Associate Publisher
Michael Swaine
Assistant *Sara Noah Ruddy*

Dr. Dobb's Journal of Software Tools (USPS 307690) is published monthly by M&T Publishing Inc., 501 Galveston Dr., Redwood City, CA 94063; (415) 366-3600. Second-class postage paid at Redwood City and at additional entry points. *DDJ* is published under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a nonprofit corporation.

Article Submissions: Send manuscripts and disk (with article and listings) to the Associate Editor.

***DDJ* on CompuServe:** Type GO DDJ

Address Correction Request: Postmaster: Send Form 3579 to *Dr. Dobb's Journal*, P.O. Box 27809, San Diego, CA 92128. **ISSN 088-3076**

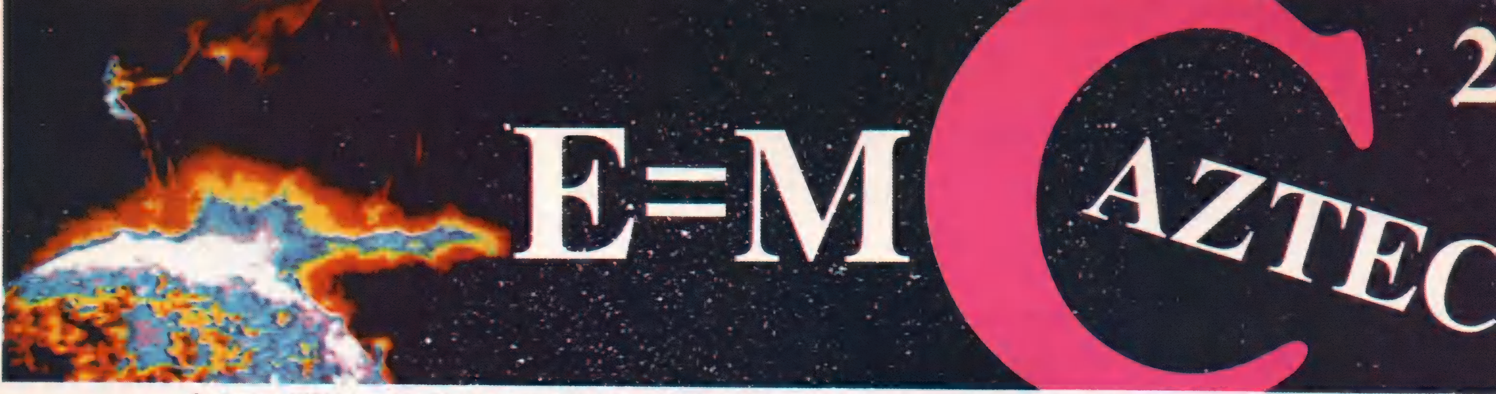
Customer Service: For subscription problems call: outside CA (800) 321-3333; in CA (619) 485-9623 or 566-6947. For book/software order problems call (415) 366-3600.

Subscriptions: $29.97 per 1 year; $56.97 for 2 years. Canada and Mexico add $27 per year airmail or $10 per year surface. All other countries add $27 per year airmail. Foreign subscriptions must be prepaid in U.S. funds drawn on a U.S. bank. For foreign subscriptions, TELEX: 752-351.

Foreign Newsstand Distributor: Worldwide Media Service Inc., 386 Park Ave. South, New York, NY 10016; (212) 686-1520 TELEX 620430 (WUI).

Entire contents copyright © 1987 by M&T Publishing, Inc., unless otherwise noted on specific articles. All rights reserved.

M&T Publishing Inc.
Chairman of the Board *Otmar Weber*
Director *C.F. von Quadt*
President and Publisher *Laird Foshay*



Our thanks to NASA for supplying this computer enhanced ultraviolet photo taken by Skylab IV of a solar prominence reaching out 350,000 miles above the sun's surface

Genius Begins With A Great Idea ...

But The Idea Is Just The Beginning

What follows is the time consuming task of giving form and function to the idea.

That's why we concentrate on building into our software development systems functions and features that help you develop your software ideas in less time and with less effort.

We've started 1987 by releasing new versions of our MS-DOS, Macintosh, Amiga, ROM, and Apple // C development systems. Each system is packed with new features, impressive performance, and a little bit more genius.

FREE 2-DAY DELIVERY!

Aztec C86 4.1

New PC/MS-DOS • CP/M-86 • ROM

Superior performance, a powerful new array of features and utilities, and pricing that is unmatched make the new Aztec C86 the first choice of serious software developers.

Aztec C86-p Professional System $199
• optimized C with near, far, huge, small, and large memory + Inline assembler + Inline 8087/80287 + ANSI support + Fast Float (32 bit) + optimization options • Manx Aztec 8086/80x86 macro assembler • Aztec overlay linker (large/small model) • source level debugger • object librarian • 3.x file sharing & locking • comprehensive libraries of UNIX, DOS, Screen, Graphics, and special run time routines.

Aztec C86-d Developer System $299
• includes all of Aztec C86-p • Unix utilities make, diff, grep • vi editor • 6 + memory models • Profiler.

Aztec C86-c Commercial System. $499
• includes all of Aztec C86-d • Source for library routines • ROM Support • CP/M-86 support • One year of updates.

Aztec C86 Third Party Software

A large array of support software is available for Aztec C86. Call or write for information. The following is a list of the most requested products: • Essential Graphics • C Utility Library • Curses • Greenleaf Communication, General, and Data Window • Halo • Panel+ • PC-lint • PforCe • Pre-C • Windows for C • Windows for Data • C terp • db_Vista • db-Query • Phact • Plink-86 Plus • c-tree • r-tree • Pmate

CP/M • TRS-80 • 8080/Z80 ROM

C compiler, 8080/Z80 assembler, linker, librarian, UNIX libraries, and specialized utilities.

Aztec C II-c (CP/M-80 & ROM). $349
Aztec CII-d (CP/M-80) $199
Aztec C80 (TRS-80 3&4) $199

Aztec C68k/Am 3.4

New Amiga Release

Amiga user groups across the USA voted Aztec C68k/Am release 3.3 the best Software Development System for the Amiga. Release 3.4 is more impressive.

Aztec C68k/Am-p Professional $199
A price/feature/performance miracle. System includes: optimized C • 68000/680x0 assembler • 68881 support • overlay linker • UNIX and Amiga libraries • examples.

Aztec C68k/Am-d Developer $299
The best of Manx, Amiga, and UNIX. System includes: all of Aztec C68k/Am-p • the Unix utilities make, diff, grep and vi.

Aztec C68k/Am-c Commercial $499
Aztec C68k/Am-d plus source for the libraries and one year of updates.

FREE 2-DAY DELIVERY!

Aztec C68k/Mac 3.4

New Macintosh Release

For code quality, reliability, and solid professional features, Aztec C for the Macintosh is unbeatable. This new release includes features and functions not found in any other Macintosh C development system.

Aztec C68k/Mac-p Professional $199
• optimized C • 68000/680x0 assembler • 68881 support • overlay linker • UNIX and Macintosh libraries • examples.

Aztec C68k/Mac-d Developer. $299
The best of Manx, Macintosh, and UNIX. System includes: all of Aztec C68k/Am-p • the Unix utilities make, diff, grep • vi editor.

Aztec C68k/Mac-c Commercial $499
Aztec C68k/Am-d plus source for the libraries and one year of updates.

Aztec C65

New ProDOS Release

Aztec C65 is the only commercial quality C compiler for the Apple II. Aztec C65 includes C compiler, 6502/65C02 assembler, linker, library utility, UNIX libraries, special purpose libraries, shell development environment, and more. An impressive system.

Aztec C65-c Commercial $299
• runs under ProDOS • code for ProDOS or DOS 3.3

Aztec C65-d Developer. $199
• runs under DOS 3.3 • code for DOS 3.3

Aztec ROM Systems

6502/65C02 • 8080/Z80 • 8086/80x86 • 680x0

An IBM or Macintosh is not only a less expensive way to develop ROM code, it's better. Targets include the 6502/65C02, 8080/Z80, 8086/80x86, and 680x0.

Aztec C has an excellent reputation for producing compact high performance code. Our systems for under $1,000 outperform systems priced at over $10,000.

Initial Host Plus Target $750
Additional Targets $500
ROM Support Package $500

Vax, Sun, PDP-11 ROM HOSTS

Call for information on Vax, PDP-11, Sun and other host environments.

C' Prime

PC/MS-DOS • Macintosh
Apple II • TRS-80 • CP/M

These C development systems are unbeatable for the price. They are earlier versions of Aztec C that originally sold for as much as $500. Each system includes C compiler, assembler, linker, librarian, UNIX routines, and more. Special discounts are available for use as course material.

C' Prime $75

Aztec Cross Development Systems

Most Aztec C systems are available as cross development systems. Hosts include: PC/MS-DOS, Macintosh, CP/M, Vax, PDP-11, Sun, and others. Call for information and pricing.

How To Become An Aztec C User

To become a user call 800-221-0440. From NJ or international locations call 201-542-2121. Telex: 4995812 or FAX: 201-542-8386. C.O.D., VISA, MasterCard, American Express, wire (domestic and international), and terms are available. One and two day delivery available for all domestic and most international destinations.

Aztec C is available directly from Manx and from technically oriented computer and software stores. Aztec Systems bought directly from Manx have a 30 day satisfaction guarantee.

Most systems are upgradable by paying the difference in price plus $10. Site licenses, OEM, educational, and multiple copy discounts are available.

MANX

CIRCLE 108 ON READER SERVICE CARD

Manx Software Systems
1 Industrial Way, Eatontown, NJ 07724

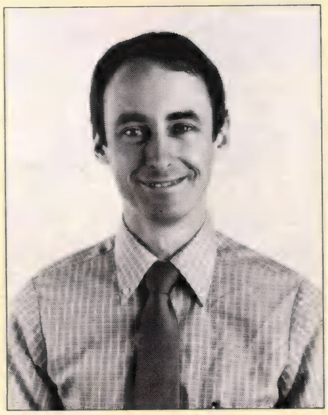
To order or for more information call today.

1-800-221-0440

In NJ or international call (201) 542-2121 • TELEX: 4995812

MS is a registered TM of Microsoft, Inc., CP/M TM DRI, HALO TM Media Cybernetics, PANEL TM Roundhill Computer Systems, Ltd., PHACT TM PHACT Assoc., PRE-C, Plink-86, Plink-86 + , P-Force TM Phoenix, db Vista TM Raima Corp., C-terp, PC-lint, TM Gimpel Software, C-tree TM Faircom, Inc., Windows for C, Windows for DATA TM Creative Solutions, Apple II, Macintosh TM Apple, Inc., TRS-80 TM Radio Shack, Amiga TM Commodore Int'l., Unix TM AT&T, Vax TM DEC, Aztec TM Manx Software Systems.

RUNNING LIGHT



In my first column I confessed to not being much of a language zealot. Though I do enjoy playing around with Forth, I don't have the religious fervor that seems to afflict so many Forth enthusiasts. I will, however, also confess to not being able to deny myself the pleasure of baiting C enthusiasts about how wonderful Forth is so that I can watch their faces turn all those pretty colors. (Forthians aren't the only people who can be fanatic about their programming language.) I will even, on occasion, dirty my hands with a program written in Turbo Pascal or QuickBASIC.

From what I've learned in talking with *DDJ* readers, this isn't all that unusual. Most of you seem to be rather fond of programming in C on MS-DOS machines, but that isn't all that surprising given the current popularity of C and *DDJ*'s history of exploring C compilers from the inside out. In fact, one of the distinguishing characteristics of a long-time *DDJ* reader is familiarity with a variety of languages.

It is with this in mind that we put together every issue of *DDJ*. Thus, you'll find this issue—our "annual Forth issue"—not completely overrun with articles written about Forth. True, there a couple of rather good Forth articles, and we're introducing a new Forth column, but there is plenty more here as well—and our coverage of Forth doesn't begin and end with this issue.

Because I'm in a confessing mood, I'll unpack another: I am a man without a number. I am neither a sixer nor an eighter. I learned to program in machine language on the RCA 1802 (I learned enough to know better—please don't send in any 1802 articles) and moved through CP/M to MS-DOS along with the rest of you. I've seen enough machines and nifty software to realize that short of having a basement with a Cray and about a dozen other computers hooked up to a magic terminal, there's simply no way I'll ever be satisfied with a single machine.

Which brings us, in a not entirely circuitous manner, to this month's lead article. Rather than putting off Rich Brown and Steve Legitt's piece on extending the AppleTalk network until January's "annual 68K issue," we're printing it now for those of you who (like me) don't want to wait for a good article. As you might expect from my comment on numbering, this is hardly the last Mac article you'll be seeing in *DDJ*. In the future, who knows? A column on Mac topics? We'll see.

One last thing. No Running Light would be complete without soliciting reader feedback and articles. What are you interested in reading in 1988? We're looking for more articles on debugging, object-oriented programming, and real-time programming for the first quarter. Call me at (415) 366-3600 or catch me on line and tell me what you'd like to see in the magazine.

Tyler Sperry

Tyler Sperry
editor

ARCHIVES

Ten Years Ago In *DDJ*

"When faced with the choice of paying $100 or more for a piece of commercial software without source code, or a freebie with source code, most people will try the freebie first. If it doesn't work out, they can always buy the commercial product.

"Personally, I wouldn't want a piece of sourceless software as a gift, let alone pay money for it. Sooner or later, I want to make modifications. Without the source code this can be a real hassle. It can be a hassle even with the source code and full documentation. . . .

"By refusing to sell source code, vendors are cutting down their sales potential. The practice can't prevent anyone from making a tape copy for his buddy down the street, nor can it keep anyone from using a disassembler to see what makes a program tick. Yet some vendors are so secretive they won't even sell you a Basic manual unless you buy the whole package and sign a non-disclosure agreement. Who needs that kind of trip?" *Jim Day, letter to the editor,* DDJ, *October 1977.*

Elegant Efficiency

"Computer programming is a form of art, far from being a discipline of science or engineering. For any specified programming problem, there are an infinite number of solutions, entirely dependent upon the programmer as an artisan. We can, however, rate a solution by its correctness, its memory requirement, its execution speed, and other qualities.

"For some applications, best is whatever is shortest and fastest. The only way to achieve this goal is to use the computer with an instruction set optimized for the problem. Optimization of the computer hardware is clearly impractical because of the excessive costs.

"Thus, one would have to compromise by using a fixed, general purpose instruction set offered by a real computer or a language compiler. To solve a problem with a fixed instruction set, one has to write programs to circumvent the shortcomings of the instruction set.

"The solution in FORTH is not achieved by writing programs but by creating a new instruction set in the FORTH virtual computer. The new instruction set in essence becomes 'the' solution to the programming problem." *C. H. Ting, "Formal Definition of FORTH,"* DDJ, *February 1982.*

DR. DOBB'S JOURNAL of COMPUTER Calisthenics & Orthodontia
Running Light Without Overbyte

Optimizing ROMable C

With Source Level Debugging on Your Target System!

Optimum-C, the compiler known for its tight, highly-optimized code generation, produces completely ROMable code. Coupled with ROM support tools like ROMable standard function libraries, a remote debugger, pre-written startup routine, interactive editor, and UNIX-style MAKE, Optimum-C will help you complete your project in record time.

Optimum-C, the ROM developer's choice

Optimum-C has the features ROM developers require in a C compiler. These features include fast execution, multiple memory support, and 8087 support via inline code or emulation. Optimium-C also has understandable error messages, one-step compiling, complete function prototyping, and inline 8086 I/O instructions.

You've seen the ads: Datalight challenges Microsoft. Our C compiler expert Richard Relph saw the ads and sent for Datalight's compiler. What he found when he began to test it must have given him mixed feelings. For the past two years Richard has been involved in developing the DDJ suite of benchmarks for C compilers. The Datalight compiler flattened those benchmarks, making them worthless.

With the apparent glut of C complier suppliers vying for the MS-DOS market, it was only a matter of time before one of them decided to step above the crowd and provide a reliable optimizing C compiler. Datalight beat all others to the punch by delivering such a compiler February of this year. (Dr. Dobbs Journal, August 1987)

ROM-*it*, The ROM Developer's Kit

ROM-*it* adds the functionality required by serious ROM developers. **ROM-*it*** includes pre-written start-up code for an embedded system to take the 8086 processor from a system RESET to executing C code. The library included has standard C functions that are usable in a ROM environment, without any hidden calls to MS-DOS or the BIOS. Lastly, the BLAZE Locator/Intel hex file generator performs location of code and data while it checks for ROM overflows, illegal data access, and complete program location.

Remote-DSD Source Level debugging on your target system!

The Remote-DSD debugger allows you to debug your application on the target system. Remote-DSD runs on the PC and is connected to your target system via an RS232 link. With Remote-DSD you download your application to the target hardware, modify initial values of variables before you start your application. The C source code appears on the PC showing you where execution will start. You can now execute or single step through your program, set break points, and access your target machine internals like I/O ports, registers, and memory.

And lastly, Support!

The product is only as good as the support. With DATALIGHT, you are covered here. You will get one hour of phone support for those needed answers. Also, you will get updates for one year, so you are always running with the latest versions.

Pick up the phone and gain a powerful advantage.

Call

1-(800) 221-6630

for complete details on how you can finish those ROM projects on time and without the bugs!

Magazine Reviewers Shocked by DATALIGHT's Performance...

"Reviewing this compiler was quite a surprise for us. For such a low price, we were expecting a "lightweight" compiler. What we got was a package that is as good as or better than most of the "heavy-weights." Datalight C implements a complete C language. It also compiles quickly, doesn't take up much disk space, and looks impressive in the benchmarks."

DR. DOBBS, August 1986

"This is a sharp compiler!... what is impressive is that Datalight not only stole the compile time show completely, but had the fastest Fibonacci executable time and had excellent object file sizes to boot!"

COMPUTER LANGUAGE, February 1986

Specifications

- Full UNIX V C compiler with ANSI extensions
- Global optimizations using Data Flow Analysis
- 8087/80287 support inline or emulation
- Complete library source code
- Multiple memory model support
- Interrupt handling in C
- Make utility
- EZ editor
- DLC one step compiling
- Start-up code for 8086
- ROMable library (without hidden MS-DOS calls)
- BLAZE locator/Intel hex file generator
- Source level debugging on the target system
- View source code as it executes on target
- Access local variables
- View 8086 machine internals

Datalight

17505-68th Avenue NE, Suite 304
Bothell, Washington 98011 USA
(206) 367-1803

CIRCLE 203 ON READER SERVICE CARD

Texas Instruments has system developers need.

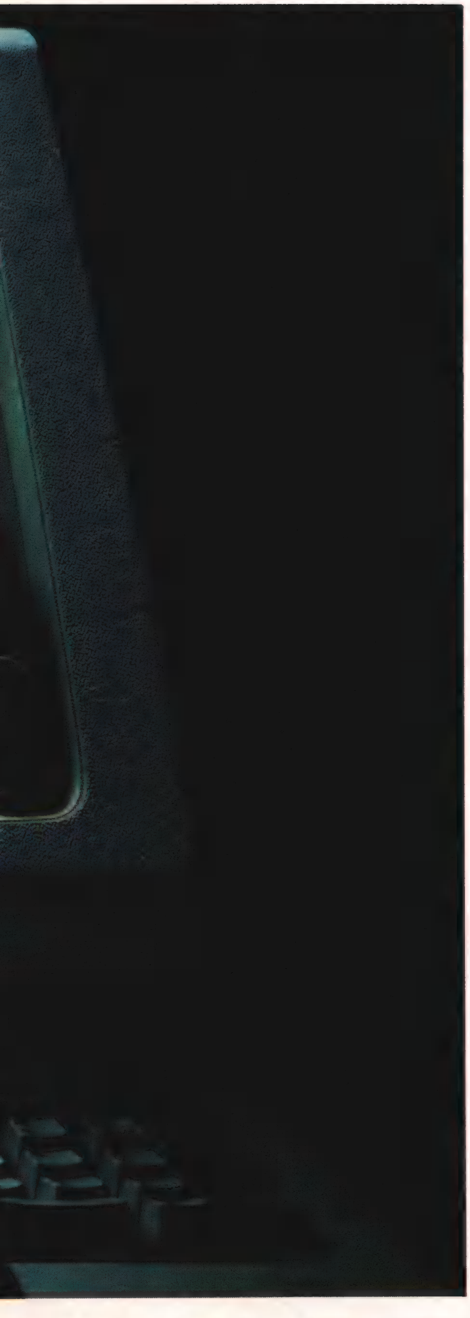
"Personal Consultant™ Plus . . . offers a very fine expert system development and delivery tool that already has a proven record with end-users."

— *Susan Shepard, AI Expert*

Personal Consultant Plus 3.0 Standard Features
- Frames, rules, meta rules and procedures
- Forward/backward chaining
- Confidence factors
- Regression testing and rule tracing
- End-user explanation facilities
- Graphics image capture and display
- Interfaces to dBase™, Lotus 1-2-3™, DOS files, .EXE or .COM programs, "C"
- Complete LISP development environment
- 2-megabyte expanded/extended memory support
- Mouse support
- Context sensitive help
- "Getting Started" tutorial-style manual

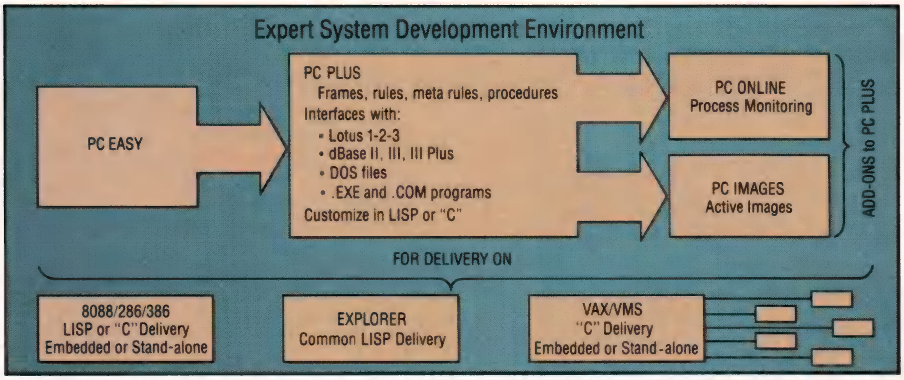
Personal Consultant Images
- Optional add-on package to PC Plus (3.0)
- Allows integration of "active images" into

what serious expert
Power tools.



Among all the expert system development tools available for personal computers today, none deliver the power and flexibility of TI's Personal Consultant series.

Personal Consultant Easy is ideal for getting started, and is upwardly compatible with the higher functionality of PC Plus. For experienced developers, Personal Consultant Plus and its optional add-on enhancements, Online and Images, were designed to help solve a broader range of complex problems.



Personal Consultant Plus. Full power for an affordable price.
At $2,950, PC Plus has proven to be one of the richest and most flexible problem-solving tools available for the development of complex knowledge-based systems. Designed to take advantage of today's more powerful 286/386 DOS-based computers, or TI's Explorer™ Symbolic Processing System, the new 3.0 version of PC Plus provides powerful standard features and a continuing growth path with the addition of either PC Images or PC Online, or both.

Personal Consultant Images. Picture an expert system with interactive graphics.
At $495, PC Images enables developers to create knowledge-based applications that incorporate complex graphical "active images." User-interactive dials, gauges, forms and selection images provide a more exciting visual data input and output style.

Personal Consultant Online. The expert system as part of the process.
At $995, PC Online allows the developer to design expert systems which interact directly with process data, as opposed to input from a human operator. Designed for intelligent process monitoring applications, this optional package helps deliver expertise that is "online all the time."

Application delivery as flexible as the tools themselves.
Delivery can be in LISP for flexibility, or "C"* for maximum speed and portability. Our "C" options support either stand-alone or "embedded" knowledge bases. Options are available for DOS-based PCs, TI's Explorer, and DEC's VAX™ line of multi-user minis running under VMS™.

"Texas Instruments has done more than any other company to educate people about AI, to popularize it, and to make useful AI tools available at reasonable prices."
— Jim Seymour, PC Magazine.

Technical support, training courses and Knowledge Engineering Services are available for the Personal Consultant products. If you have a question about any of our expert system power tools, we have the answer.

Pick up the phone and gain a powerful advantage.
Call **1-800-527-3500** for technical overviews of our products and a PC Plus case histories brochure which details how our power tools are being put to work today.

knowledge bases
- Interactive dials, gauges, forms and selection images
- Multiple images can be combined on same screen
- "Getting Started" tutorial-style manual

Personal Consultant Online
- Optional add-on package for PC Plus (3.0)
- ONLINE expert systems that interact directly with process data
- Multiple interfaces to data acquisition and analysis programs
- Knowledge base synchronization with process data
- Functions for historical and predicted trends
- Special user interface/reporting capabilities
- "Getting Started" tutorial-style manual

36106
© 1987 TI
Personal Consultant and Explorer are trademarks of Texas Instruments Incorporated.
dBase is a trademark of Ashton-Tate.
Lotus 1-2-3 is a trademark of Lotus Development Corp.
VAX and VMS are trademarks of Digital Equipment Corporation.

*Available 4Q 1987.

TEXAS INSTRUMENTS

LETTERS

A-to-D Conversion

Dear *DDJ*,

I have some comments on John Musselman's article on page 22 of the June issue. All but the first are Complaining Songs, but remember the first *is* first because it's important. I would not complain about details if I didn't like what the author was doing.

1. The circuit in Figure 1 and the insight that it suffices to digitize voltages are delightful. I can only gasp "Nifty!" and "Why didn't I think of that?"

2. The proof of a pudding, however, is in the eating. I want to see calibration data before I believe it's so easy. Some of the problems I anticipate are saturation effects when the unknown voltage gets close to the limits (for example, 0 or +5 volts) and interrupt latency jitter. I don't know how your TMS7000 works, but the specs for my Z80A seem to say that the delay from the interrupt signal to start of the interrupt routine can vary by 5.8 microseconds, depending on what the CPU is doing when the interrupt comes. I guess it could be even longer if other interrupts were active. The averaging process would tend to wash out any effect, but I would try some worst-case tests to see if it causes any trouble.

3. Whenever the input voltages at the comparator are not (nearly) equal, the Out bit stream will be solid 1s or 0s until the voltages become equal. These bits don't belong in your average, so you must—if there is any chance that an average will be asked for before comparator voltages get equalized—delay accumulation of the average until at least one transition 0<==>1 comes down the bit stream. I don't see this safety feature in the software, but it would be needed, for example, for a time multiplex application, in which a sequence of voltages was measured one after the other. When John Musselman proposes a digitizer using only one I/O bit, I am impressed, but when I see it needs more than a second per channel in time multiplex, I start to nod off.

Why not go at it like this? Let the time constant, RC, be much longer than the period, P, between interrupts (for example, 1 millisecond). Assume you know the voltage, $V(T)$, on the capacitor at time T, when an Out=1 period begins. By Ohm's law the current into C is $I(T)=(H-V(T))/R$, where H is the logic 1 voltage. Because $P<<RC$, the total charge flow during this interrupt period is very nearly $P(H-V(T))/R$, and the voltage at T+P is, to first order in P, $V(T+P)=V(T)+P(H-V(T))/(RC)=V(T)(1-P/(RC))+HP/(RC)$. If, instead, a 0 is sent out, $V(T+P)=V(T)(1-P/(RC))+LP/(RC)$, where L is the logic 0 level.

In words, the voltage on C at the end of a period is the average of the voltage at the start of the period, weighted by $1-P(RC)$, and the voltage Out, weighted by $P/(RC)$. Repeating the procedure, you can calculate $V(T+2P)$ and so on.

The "old" voltage is always multiplied by $(1-P/(RC))$. This is a "forgetting factor" because it is close-to-but-rigorously-less-than 1. (The arithmetic needs some thought lest truncation error obscure this fact.) So, even if you use a wrong number for $V(T)$, the error gets squashed down by one forgetting factor at each interrupt; $V(T)$ and the actual voltage on C get ever closer. But you don't need to guess $V(T)$: initialize Out to logic 0 before loading the program, and start interrupts at T=0 with the (excellent) assumption that $V(0)=L$.

Meanwhile it's still true that a bit stream transition 0<==>1 marks a voltage crossover at the comparator inputs, so the current value of $V(T)$ is already a good estimate of the unknown voltage; nothing is gained by averaging for another second.

Arthur's entry into software piracy was marred somewhat by his lack of professional discretion.

Thus, in a given setup, John Musselman's method (usually called the boxcar average) and the forgetting-factor average converge to the same result on a DC voltage. But when the boxcar algorithm is ready to start measuring, the forgetting-factor algorithm already has the answer.

The pudding is still pie in the sky. I will not be able to run the necessary tests soon, but I hope that someone else will do so and send me the conclusions.

R. W. Hartung
408 Orchard
East Lansing, MI 48823

Curses

Dear *DDJ*,

Thanks to Allen Holub for his article on curses [C Chest, July 1987]. Last year I, too, needed a curses package to assist in

The Leaders Made PVCS
The Leading Source Code Control System.

NOW VAX/VMS & MS-DOS Versions

When it comes to maintaining their most valuable asset, the leading software publishers rely on the POLYTRON Version Control System (PVCS). From accounting firms to airlines, the leading service companies depend on PVCS to maintain the integrity of their programs. Leading manufacturing companies use PVCS to maintain their state-of-the-art software. Leading high technology companies turn to PVCS to handle configuration management for software projects that represent an investment of hundreds of thousands of dollars. The largest aerospace companies and defense contractors use PVCS to maintain integrity of projects during development and after delivery of software. Independent programmers use PVCS to improve their productivity and software quality for themselves and their clients.

Simplify Configuration Management

When large and complex software programs are being developed on personal computers or VAX minicomputers, effective management of the revisions and versions becomes critical. PVCS simplifies this process and lets you effectively control the proliferation of code changes. We used UNIX SCCS and RCS as models. However, our own experience, and the input of hundreds of programmers and managers has enabled us to significantly improve upon these models.

PVCS provides many powerful functions including:

- Storage & Retrieval of multiple revisions of text.
- Maintenance of a complete history of changes.
- Maintenance of separate lines of development using branching.
- Merging simultaneous changes.
- Resolution of Access Conflicts.
- Modules can be retrieved by their own revision number, system version name, or specified date.
- Uses "reverse deltas" to rebuild a prior version making PVCS the fastest version control system over the project life cycle.
- Projects already under development or in the maintenance stage can be easily put under the control of PVCS.

Manages Development On Local Area Networks

Programming teams using Local Area Networks depend on PVCS to help the managers and team members work together. In fact, Novell and 3Com themselves depend on PVCS to manage the versions of their own network software products.

Supports MS-DOS and VAX/VMS Development

Now, companies that develop software on VAX systems running VMS can also use PVCS. And since the VMS and MS-DOS versions of PVCS use the same "logfile" format, you can easily develop software on PCs and maintain the code on the VAX or vice versa. The menu-driven, screen-oriented interface (and optional command-driven interface) makes it easy for programmers and librarians or administrators to use PVCS on a PC or VAX or both systems.

PVCS Maintains System Integrity

PVCS prevents corruption of code that could ordinarily result from security breaks, user carelessness or malfunctions. The levels of security can be tailored to meet the needs of your project.

PVCS & PolyMake Work Together

PolyMake, the leading MS-DOS make utility, is now available for the VMS operating system. This allows you to write makefiles that will function in both PC and VAX environments. Additionally, PolyMake reads time & date stamps of PVCS archives for fast, accurate program rebuilding.

PVCS and PolyMake Maintain Source Code Written In Any Language.

The Preferred Version Control System

The customers listed below are just a few of the innovative leaders that have made PVCS the leading version control program for personal computers.

Alcoa Aluminum
Arthur Anderson
AT&T
Ashton-Tate
Bank of America
Bell Labs
Bendix
Boeing
CIGNA
Citibank
3Com
Colonial Penn
Commerce Clearing House
Control Data Corp.
Corvus
CXI
Digital Equipment Corp.
Deloitte Haskins + Sells
Diebold
Dow
Dunn & Bradstreet
EDS
Educational Testing Service
E-Systems
Equitable Life
Federal Express
First Boston
Ford
Fox Software
Fujitsu
GTE
Hardees
Hewlett-Packard
Honeywell
Hughes Aircraft
IBM
Industrial Networking
Intel
ISC Aerospace
IVAC
Javelin
Lattice
Lawrence Livermore
Lotus
McData Corp.
McDonnell Douglas
Mead Data Central
MIT Lincoln Labs
Nastec
Novell
NCR Technologies
Pitney Bowes
Plexus Computers
Price Waterhouse
ROLM
Rockwell International
Safeco
Sears
Security Pacific
Sperry
Software Publishing
Spacelabs
Standard Oil
Standard & Poors
Tandem
Tektronix
Telex
Texas Instruments
Touche Ross
Unisys
United Airlines
United Parcel Service
United Technologies
U.S. West
Westinghouse Electronics
Xerox

Only PVCS meets the needs of independent programmers and corporations. Once you standardize on PVCS, the archives used to track and monitor changes are interchangeable between any PVCS product. You will receive full credit for your initial purchase if you upgrade to a higher-priced MS-DOS version of PVCS.

Personal PVCS — Offers most of the power and flexibility of Corporate PVCS, but excludes the features necessary for multiple-programmer projects.

Corporate PVCS — Offers additional features to maintain source code of very large and complex projects that may involve multiple programmers. Includes multi-level branching to effectively maintain code when programs evolve on multiple paths (e.g. new versions for different host systems, or a new program based on an existing program).

Network PVCS — Extends Corporate PVCS for use on Networks. File locking and security levels can be tailored for each project.

PVCS for VAX systems — Requires VMS. Uses the same interface and archive format as MS-DOS version. Supports branching and offers file locking and other security features for multiple-programmer projects.

	MS-DOS*	VMS		
	PC/XT/AT	Micro VAX II	VAX 7xx	VAX 8xxx
Personal PVCS	$149			
Corporate PVCS	$395			
Network PVCS	$995**	$4,950	$9,500	$10,500+
PolyMake	$149			
Network PolyMake	$447**	$1,250	$2,375	$2,500+

*Compatible with MS-DOS 2.0 through 3.3. Compatible with the IBM PC/XT/AT & other MS-DOS PCs.

**5 Station LAN License. Call for pricing on larger Networks.

TO ORDER:
VISA/MC 1-800-547-4000.
Dept. No. 355
Oregon & Outside USA call (503) 645-1150.
Send Checks, P.O.s to: POLYTRON Corporation, 1815 NW 169th Place, Suite 2110, Beaverton, OR 97006.

POLYTRON

High Quality Software Since 1982

CIRCLE 283 ON READER SERVICE CARD

(continued from page 12)

porting code between my PC and Unix, so I ended up writing my own. Since then, my package has been expanded significantly (it now provides almost full Unix System 5 compatibility). After receiving several requests for it, I decided to distribute my package (now called PC Curses) as shareware. The package is compatible with Microsoft C 4.0 and includes small- and large-model libraries. If any *DDJ* readers are interested in obtaining a copy, they can do so by sending me one 5¼-inch floppy along with a mailer, return label, and postage (or $5). Readers outside the U.S. should write for information.

Just a few comments/clarification about the article. After complaining about the missing *vsprintf* function, Holub may be happy to hear that *vsprintf* is beginning to appear on various Unix systems (I know that Unix System 5 has it). His comment about using *nl* (instead of *nonl*) seems logical, but Unix curses encourages the latter; moreover, some implementations of Unix curses fail to work properly unless *nonl* is specified. The comment that "... the *refresh()* command only works under the real curses if all windows are subwindows of *stdscr*" is incorrect if it is referring to subwindows created by *subwin()*; I suggest that readers forget they ever read that sentence. Also, Holub made several other comments about subwindows that are incorrect or confusing, and I suggest that anyone using other versions of curses consult the documentation (or code) for more information.

Jeff Dean
710 Chimalus
Palo Alto, CA 94306

Educatin' Programmers

Dear *DDJ*,
I wish to comment on the letter from Neil Pignatano published in your July 1987 issue.

I agree with Mr. Pignatano's assertion that mathematical reasoning is the best paradigm for passing on critical analytical skills to students of computer programming and is certainly superior in this regard to classical languages, as suggested by Allen Holub in his Viewpoint of April 1987.

I most stringently disagree, however, with the point Mr. Pignatano makes next—that there are enough "text editors, database managers, and so on." I believe that the era of truly productive productivity programs is just beginning to emerge and that it is precisely this field that will offer the most challenges and opportunities for programmers in the future.

Mr. Pignatano's views on "electronic desk" software are well known (to his coworkers), but I feel that he is missing out on the possibilities that are continually offered by new developments in hardware technologies; Neil may be content using TECO for his text editor, but I certainly am not!

Mr. Pignatano, as a physicist and a scientific programmer by profession, perhaps does not appreciate the degree to which difficulty of use and obtrusiveness (I might say "pig-headedness") of most currently available software tools inhibits the application of computer solutions to many tasks in business, education, and even technical environments. Unless text editors, database managers, and spreadsheet software begin to respond in consistent and predictable (to the new and occasional user) ways, microcomputers will fail to reach a significant fraction of their potential markets.

Naturally, this belief of mine leads me to conclude differently from Mr. Pignatano on the place of scientific programming in the education of programmers. Rather than the strong emphasis that Mr. Pignatano would have it given, I feel that the overhead intrinsic in the explanation of scientific problems for computer solution make them more appropriate for only occasional use in computer science curricula. I maintain, rather, that scientists of all sorts should be given a strong background in computer programming techniques that they will be able to apply to the endeavors of their own disciplines—a point on which I sense that Neil Pignatano and I can agree.

Martin Veneroso
1646 Latham, #10
Mountain View, CA 94041

VM/RUN Update

Dear *DDJ*,
Your readers may be interested in the following updates to Richard Relph's review of VM/RUN from Softguard Systems [see the July 1987 issue]. The February version, which he reviewed, has been superseded by Release 1.10, which is now available.

The 7 to 15 percent penalty in execution time has been totally eliminated. The problem was caused entirely by diagnostic code that kept the "global exact" flag on at all times. This resulted in instruction fetch degradation. Subsequent benchmarks (with test problems provided by your reviewer) have shown comparable results between the VM/RUN environment, which runs at CPL 3, and Phar Lap's, which runs at CPL 0.

Full directory path support is now available. Support files no longer have to be in the current directory, and the number of files has been reduced.

Significant improvements have been made in application load times. Large-scale applications of ½ megabyte in size are typically loaded in about 5 seconds on a Compaq Deskpro 386.

Finally, let me say that, with the addition of a symbolic debugger and new features, such as the capability of calling 8086 code from a 386 program and vice versa, the VM/RUN environment offers software developers a powerful tool for the creation of 386 applications.

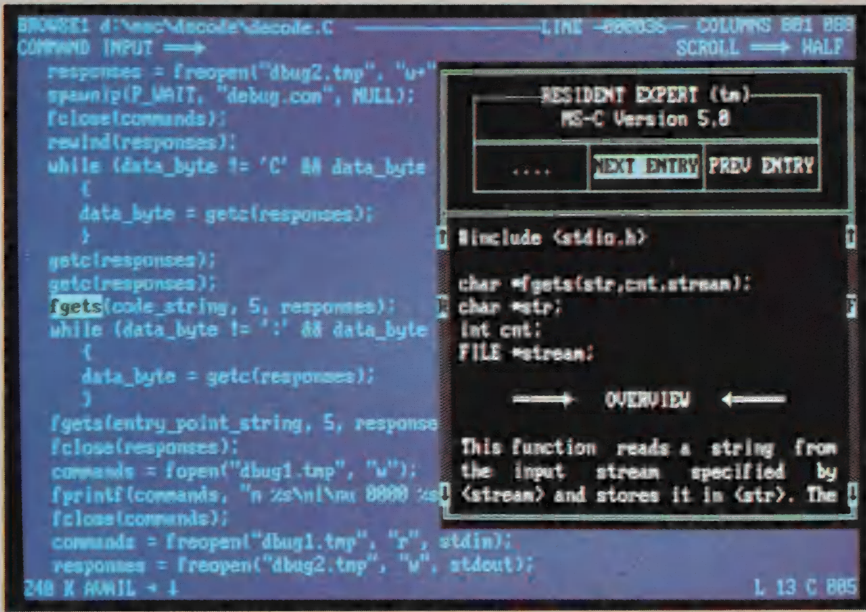
Ken Williams
Softguard Systems Inc.
2840 San Tomas Expy., Ste. 201
Santa Clara, CA 95051

The Problem Is Architecture

Dear *DDJ*,
Frank Albe's letter in Ray Duncan's July 1987 16-Bit Software Toolbox column doesn't get to the point about the computer language controversy. The problem is not with the languages but with the computer architectures that (do not) support them.

Computers are integer manipulators with simple decision-making capabilities. But the world is not com-
(continued on page 144)

RESIDENT EXPERT Pop-up Reference Guides...



Try One And Get Our MS-DOS/PC-DOS Guide Absolutely FREE!

THE POP-UP REFERENCE REVOLUTION BEGINS

How much development time could you save if you never had to open another PC language or technical reference manual again? What if you could just point at a compiler keyword, assembly instruction, or function name *on your screen* and with a keystroke have complete, authoritative information about language syntax, operands, parameters, examples, and much more?

INTRODUCING THE RESIDENT EXPERT SYSTEM

A growing library of comprehensive, disk resident reference guides about the PC *and* your favorite PC languages. All available instantly through our unique memory resident pop-up access system.

VIRTUALLY EVERYTHING YOU NEED TO KNOW

Each of our *Compiler Reference Guides* contains virtually everything you need to know to program with your *preferred implementation* of your favorite language. Language syntax, all library functions, compiler directives, and error codes are thoroughly documented.

Our *PC Programmer's Reference Guide* documents every PC (and AT) processor instruction and every BIOS and DOS service interrupt. You'll also find tables of keyboard codes, line drawing, ASCII, and IBM character sets, and much more.

THE SPECIALIST'S LIBRARY

Your compiler is unique. That's why our reference guides are *specialized*...each one designed for a particular vendor's language implementation.

Free With Any Purchase!

Our Companion Pop-up Guide To MS-DOS 3.2/PC-DOS 3.3

Limited Time Offer

QUICK DRAW ACCESS SYSTEM

Point-and-shoot...just place the cursor over any term on your screen. Chances are we've got it fully detailed in one of our data bases.

Fully cross indexed...if the instruction or library function you're using isn't quite right, our related topics cross index can help you find a better one.

Multiple volumes on line...you can have one or a dozen of our pop-up reference guides on line...a complete library available instantly.

THE INFORMATION YOU NEED...WHERE YOU NEED IT

Our pop-up shell varies its size and shape dynamically, only taking as much space on your screen as it needs and it *never* covers your working area. You can see your work and our reference data at the *same* time.

A COMPLETE LIBRARY...STILL ONLY A BEGINNING

At Santa Rita, our commitment is to provide the most accurate, extensive selection of PC language reference materials available. If you don't see one of our guides for your favorite language or compiler listed below don't worry, we're probably working on it!

PC Programmer's Reference Guide . . . $59.00
(with Assembly Language Guide)

Borland Turbo C *(1.0)* 59.00
Borland Turbo Pascal *(3.0 and below)* 59.00
(with Graphics & Numerical Methods Toolbox)
Borland Turbo Prolog *(1.1 and below)* 59.00
(with Prolog Toolbox)
Lattice C Compiler *(3.2 and below)* 59.00
Mark Williams LetsC *(4.0 and below)* 59.00
Microsoft C Compiler *(5.0 and below)* 59.00

SantaRita

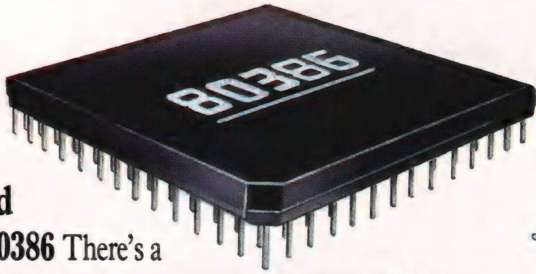
For the location of your nearest Santa Rita Software dealer, or to order direct, call us at 1-214-727-9217. We'd like to hear from you.

Santa Rita Software
1000 E. 14th Street, Suite 365
Plano, Texas 75074

The RESIDENT EXPERT System

Resident Expert is a trademark of The Santa Rita Company. Borland, Turbo C, Turbo Pascal, and Turbo Prolog are trademarks of Borland International Inc. IBM and PC-DOS are trademarks of International Business Machines Corporation. Lattice C is a trademark of Lattice Inc. LetsC is a trademark of Mark Williams Company. Microsoft and MS-DOS are trademarks of Microsoft Corporation.

A Number of Reasons A Number



1. Designed for the 80386

There's a revolution taking place in desktop computing. A revolution that's been launched by a square wafer of silicon known as the 80386 microprocessor chip. It puts minicomputer potential at PC users' fingertips. It's a fact that virtually every leading PC manufacturer has built a "box" around this chip. And it's a fact that the "New Operating System" will, supposedly, even run on it. But, it's also a fact that *their* system wasn't designed for the 80386. Ours is. And it's called PC-MOS/386.™

4. Thousands of DOS Programs

PC-MOS/386™ gives you the best of the past, and the best for your future. Which means that while PC-MOS/386™ totally replaces your old DOS, you won't have to replace the programs you've spent a lot of time learning.

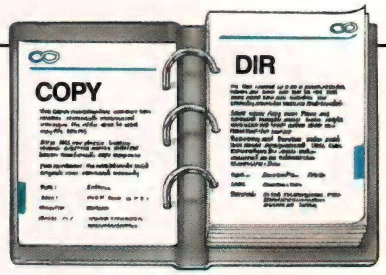
And it all happens so effortlessly. You'll continue to reap the benefits of your favorite DOS programs, while entering a new arena of power.

Think of it! Programs like dBASE III, WordPerfect, Lotus 1-2-3 and Symphony, WordStar, MultiMate...literally thousands of DOS programs–all compatible and multi-user available.

2. PC and PS/2 Compatible

In designing PC-MOS, we knew our first priority was to exploit the minicomputer capabilities of 80386-based PCs & PS/2s. But we went further, and developed a system which would be fully compatible with the millions of existing PCs, PC ATs, and PC-compatibles. Power without sacrifice. You'd expect nothing less from the new standard bearer.

5. Familiar Commands Like DIR and COPY



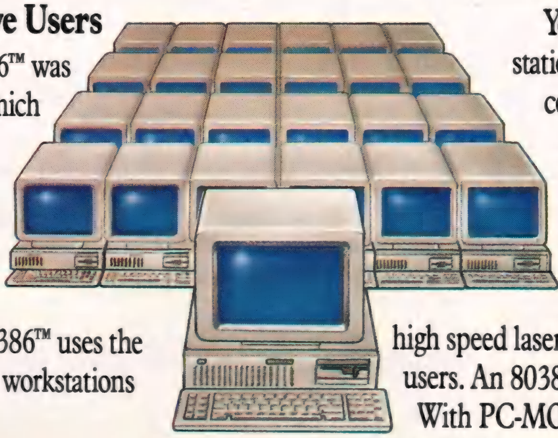
Just as you don't have to learn a whole new array of software to take advantage of PC-MOS/386,™ neither do you have to learn an entirely new set of commands.

Instead, the system builds on the knowledge you already have. "COPY" still copies files, and "DIR" still gives you a directory listing. As you might expect, we didn't stop there. There's a wealth of features that have strengthened the commands you know, making them more powerful and easier to use.

3. One, Five, Up to Twenty-five Users

From the beginning, PC-MOS/386™ was designed as a versatile operating system which could support twenty-five users as easily as it supports one. The system comes in single, five, and 25-user modules, so you're able to start with what you need and expand when you're ready.

In a multi-user setting, PC-MOS/386™ uses the computing power of the host PC to drive workstations linked to standard RS-232 ports.



You can choose from a variety of workstations. Mix and match dumb terminals costing under $500 each with PCs and PS/2s running our terminal emulation software.

All of the host's resources can be shared. Programs, data, hard disks, tape backup units & printers (including high speed laser printers) are suddenly available to all users. An 80386-PC has minicomputer potential. With PC-MOS/386™ you can "mini" your micro.

# of Users Will Choose PC-MOS/386.™

6. Concurrently Supports Virtual 8086 and 80386 32-Bit Mode

80386-based PCs & PS/2s are dual-personality computers. To run DOS programs, they act as PCs with a 640K memory limit. But to take advantage of their minicomputer capacity, they operate in true 80386 mode which lets them address up to four gigabytes of memory. PC-MOS enables the 80386-host and its workstations to independently switch between these modes–making DOS compatibility and 80386 power simultaneously possible.

7. Multi-Tasking

While it's true you could look elsewhere for multi-tasking, why would you want to? The *other* multi-tasking operating system is not now, nor is it planned to be, multi-user. It won't even run multiple DOS applications in multi-tasking mode.

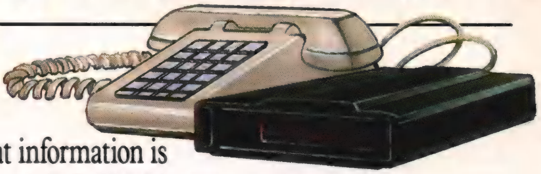
Now consider PC-MOS/386.™ At the touch of a key, you can switch between up to 25 different tasks. And if you have workstations connected to a host, they get multi-tasking, too. Finally...a system that won't hold you back.

8. File/Record Locking and Security

When you decide to implement either a network or a multi-user system, there's a two-fold problem which must be solved: protecting your work from accidental misuse and securing it from intentional theft.

PC-MOS/386™ solves both aspects of this problem. Password protected security allows you to assign file, directory, and task access to each user. Plus, files and records are locked using either PC-MOS' proprietary system or NETBIOS emulation.

9. Remote Access



It's been said that information is power...which makes PC-MOS/386™ a deadly weapon to your competition. Imagine on-the-road salespeople being able to file call reports and access your latest inventory data. Picture executives being able to access your corporate database from across the country, or around the world–giving them the information they need, when they need it.

Visualize branch offices tapping time-critical data with nothing more than a modem and a workstation. Working at a home office in the evening or over the weekend suddenly gets awfully productive. And that makes good business sense. The kind of sense you can't afford to be without.

10. The Price...

As you evaluate operating systems, ask yourself if it's reasons you're considering...or rhyme. Ask if you're getting a system for tomorrow, or one that was made for yesterday. See if you're being forced to buy new hardware because of *their* software.

And consider this.

Only one operating system in the world can give you the raw power, features, and functionality that you demand. Its name is PC-MOS/386.™ And it's immediately available in one, five and 25-user versions starting at $195.

PC-MOS/386™ is a trademark of The Software Link, Inc. PS/2, PC AT, NETBIOS, dBASE III, MultiMate, WordPerfect, Lotus 1-2-3 & Symphony, & WordStar are trademarks of IBM Corp., Ashton-Tate, WordPerfect Corp., Lotus Development Corp., & MicroPro, respectively. Prices and technical specifications subject to change. Copyright ©1987. All Rights Reserved.

For the dealer nearest you, **CALL: 800/451-LINK** | In Georgia: 404/441-2580 | International/OEM Sales: 404/263-1006 | Resellers/VARs: 404/448-5465

3577 Parkway Lane, Atlanta, GA 30092 Telex 4996147 SWLINK FAX 404/263-6474

The Software Link/Canada CALL: 800/387-0453

DEALER INQUIRIES INVITED

PC-MOS/386™

MODULAR OPERATING SYSTEM

THE SOFTWARE LINK

CIRCLE 381 ON READER SERVICE CARD

Async AppleTalk

by Richard E. Brown and Steve Ligett

The design and implementation of an RS-232 enhancement to the AppleTalk protocols

For most users AppleTalk appears to be nothing more than an expensive cable. Actually, it's much more. It's a complete family of data communication protocols designed to allow Macintoshes and other computers to share peripherals and resources. Although the initial AppleTalk service provided a printing connection between a Macintosh and an Apple LaserWriter, these days there are literally dozens of products, including other high-quality output devices and disk or file servers, that take advantage of this 230.4-kbps link between devices.

Async AppleTalk, a software application developed at Dartmouth College, was implemented to expand the potential of this network by allowing AppleTalk devices to be connected over an asynchronous (RS-232) link. The software provides low-cost, remote access to AppleTalk devices and works with nearly all AppleTalk applications.

The first step in designing Async AppleTalk was to define the necessary protocols. A protocol is a set of rules that two or more parties (human or computer) use to conduct orderly discourse. A protocol might, for example, involve two computers negotiating to send a file between them. The conversation could be as simple as that illustrated in Figure 1, page 20. A real protocol would have to specify what would happen if computer B couldn't take a file that big, or if it already had a file named FRED, or if it wasn't saved correctly at the end. But you get the picture—a few more rules could make the scheme watertight.

Richard E. Brown and Steve Ligett, Kiewit Computer Ctr., Dartmouth College, Hanover, NH 03755. Rich is manager of special projects at Dartmouth. He has been writing communications software for seven years and is a member of the IEEE and the ACM. Steve is an engineer in the special projects group of the computing services department at Dartmouth. He writes software for personal computers and designs and builds hardware for Dartmouth's local-area network.

If you were designing the program to implement this protocol, you would like the code to be about as simple as the problem definition. The top-level code should implement the rules mentioned earlier: it shouldn't have to worry about each little mishap that might occur anywhere in the transmission chain. Consequently, you would design this simple file transfer protocol as a network layer and delegate to another layer the responsibility for detecting errors, retransmitting corrupted messages, or sending messages around a network.

Network layering brings to data communications the simplicity that structured programming brings to software. Using layers designers can isolate issues and design solutions for each of the separated areas—for example, code that detects transmission errors solves a very different problem from code that implements the file transfer protocol discussed earlier.

Layering also allows substitution of equivalent services by a different implementation. The link access protocol (LAP), for example, is generally responsible for sending sequences of bytes (frames) on a wire. Different LAP software and/or hardware can use different kinds of wire. The standard AppleTalk Link Access Protocol (ALAP) layer uses twisted-pair wire. Async AppleTalk replaces that wire with a modem or RS-232 link. Another LAP replacement is an AppleTalk to Ethernet converter—it uses the 10-Mbps link to deliver AppleTalk datagrams. All three LAP layers have their place; none of them could have been done if the highest-level code had to deal with all the successive details of data transmission.

Each network layer provides a well-defined service to a

higher-level client. The client then communicates with a peer across the network. The effect is that peers appear to be talking directly to each other, as if the lower layers weren't there at all. Each layer specifies the means of communicating with its peer—the protocol—and the service it will provide to its clients.

Peers communicate by asking a lower layer to perform a service for them. The lower layer can, in turn, invoke still lower layers to do more work. The lowest layer—the physical layer—is the electronics that sends and receives the bits across a link. At the other end, each layer, starting at the lowest, passes the received data up to its client until the messages reach the peer of the originator.

Another concept in networking is the onion-skin principle. When a layer has data to communicate with its peer, it passes the data to the next lower layer. That layer adds its own information (called a wrapper) and passes the message to the next lowest layer, and so on. When the message gets to the bottom (physical) layer, it is actually transmitted. Each peer at the other end verifies any information contained in the header, removes the corresponding wrapper, and passes the resulting data up to its client—thus, the analogy to the layers of an onion. In the earlier file transfer example, the file transfer layer might send data to a layer that guarantees delivery of the information it was given. This layer could wrap routing information around the message and pass it on to the LAP layer to be sent out on the wire. Figure 2, page 20, shows these transformations.

There is an interesting symmetry in the transmit and receive algorithms. Data to be transmitted is passed down through subroutine calls from one layer to another until it is finally transmitted on the link. Just the opposite sequence occurs when a frame arrives in a computer. The receiver accepts characters until it gets a full frame. If the frame is good (CRC OK, proper length, and so on), the receiver makes a subroutine call to the next layer with the data as a parameter. Depending on how the software was set up, that layer may make further subroutine calls to its next higher layer until the message arrives at the final recipient.

For a detailed and quite readable explanation of networking, take a look at Tanenbaum's *Computer Networks*.

The AppleTalk Protocols

Each protocol of the AppleTalk family performs a specific function. The AppleTalk Transaction Protocol (ATP), for example, specifies how multiple blocks of data (a transaction of, say, eight disk blocks) can be reliably transferred between nodes (that is, Macs) on a network. The Name Binding Protocol (NBP) specifies how to convert a (text) name to a (numeric) network address. The full details of the AppleTalk protocol family are available from *Inside AppleTalk*.

Both ATP and NBP rely on the ability to send a single message through the network, possibly directing (routing) the message through multiple links. The Datagram Delivery Protocol (DDP) specifies how these messages are routed. Another protocol, the Routing Table Maintenance Protocol (RTMP), describes how to maintain tables of routing information.

All AppleTalk protocols rely on the ability to send a frame of data (an orderly sequence of 8-bit bytes) to a neighbor node. The AppleTalk Link Access Protocol (ALAP) specifies the rules for access to the twisted-pair bus, which connects devices on AppleTalk. Just as with a telephone party line, AppleTalk devices must cooperate to keep their messages from interfering. ALAP uses a three-way exchange to minimize the time wasted by collisions. The initiator of the frame listens to see if the bus is in use; if so, it waits a random amount of time and retries. If the bus is idle, the initiator sends a request-to-send (RTS) frame to the intended recipient. If the recipient is prepared to handle the data, it responds with a clear-to-send (CTS) frame. If the initiator receives the CTS, it then sends the data.

If two devices begin to transmit simultaneously, their

(continued from page 19)

RTS frames will collide, garbling both. Their recipients will not receive the RTS correctly and will not respond. Each initiator will fail to receive a CTS, wait a random amount of time, and then retry the transmission sequence. By waiting a random interval, one of the initiators will retry first and will probably succeed.

A final (and very important) point: notice that ALAP does not guarantee correct delivery of a frame. A transmission error could garble a frame; when that occurs, the recipient simply discards the frame as if it had never arrived. How does the data ever get through? Higher layers (ATP or NBP) establish rules for acknowledging messages that must be delivered.

An AppleTalk Implementation

In late 1983, Dartmouth College recommended that the freshmen entering in the fall of 1984 purchase Macintosh computers. It also decided to expand its campuswide data network to support communications in the students' rooms. At that time, Dartmouth had a network of 1,800 terminal ports (asynchronous, RS-232 hard-wired and dial-up ports) served by approximately 40 network nodes (minicomputers) that were located in the basements of academic and administrative buildings. A user at any terminal could connect to any host computer as well as to the library's on-line card catalog, a campus events calendar, or other off-campus networks.

Over the summer of 1984, AppleTalk support was added to the network. Immediately, the number of nodes was more than doubled (to a total of 95) to support the dormitories. The dorms were wired with 2,600 AppleTalk outlets, one plug per pillow. We decided not to retrofit all the async ports to AppleTalk as it would have been prohibitively expensive.

We also developed a terminal emulation program, called DarTerminal, which works across this AppleTalk network to all the host computers. DarTerminal acts as a VT-100 terminal, a TEK10 (4012) graphics terminal, and a distributed screen editor, in which the Macintosh acts as a screen manager for a back-end host editor (now available on the DCTS, VAX/VMS, and Unix systems). DarTerminal also offers simultaneous terminal sessions, cut-and-paste transfers between sessions, and file transfers of entire Macintosh documents.

Unfortunately, in that one stroke, we rendered obsolete the 1,800 asynchronous ports located in the college's academic and administrative offices. Of course, the ports still worked, but over time, as people in those offices bought Macs, they couldn't get the benefits of AppleTalk that we had already provided to the students. This caused some real frustration because faculty couldn't use the same tools as their students. Staff wanted to use a terminal program that was tuned to the students' computing environment. As network maintainers, we wanted to use the multiple sessions to troubleshoot network problems from home (instead of driving to the computer center on cold winter nights . . .).

Computer A		Computer B
Take a file called FRED of length 123456	-->	
		<-- OK.
Here it comes.	-->	
<much data flows>	-->	
That's all. Save it.	-->	
		<-- OK.
Thanks.	-->	

***Figure 1:** Two computers negotiating to send a file between them*

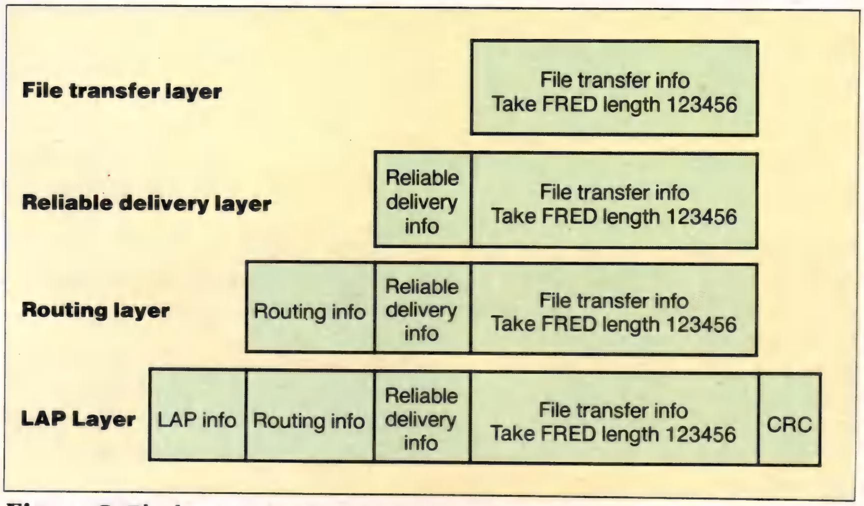


***Figure 2:** The layer onion skin*

There was no off-the-shelf way to make the async ports of the network deal with AppleTalk frames. Yet we felt it was important to let people use DarTerminal over their RS-232 ports. One solution would have been to make an "Async DarTerminal" that ran directly on an async port or modem. This might have worked, but it would have been troubled by flow control (does XOFF mean stop sending, or is it just another data character?) and we would have had problems maintaining two versions of the program. Furthermore, it would only have solved the specific problem of DarTerminal; other AppleTalk applications such as printing and file sharing wouldn't have been able to exploit that solution.

Instead, we chose to send AppleTalk frames over an async link. To distinguish it from Apple's standard link access protocol, we named it Async AppleTalk Link Access Protocol (AALAP). AALAP frames contain all the information conveyed in the 230.4-kbps frames. Furthermore, AALAP provides exactly the same software interface to the higher AppleTalk layers (DDP, NBP, and so on) in the Macintosh. In this manner, the layers above AALAP are "fooled" into thinking there is a 230.4-kbps link in

COPYPHOBIA?

Every day, PC software developers confront the problem of software piracy. Unfortunately, protection methods block even legitimate backup copying, and legally induced fear doesn't seem to stop would-be pirates. Either way, you — the publisher — lose.

But now you can stop software pirates from using your products without paying — with the DS1207 TimeKey™ from Dallas Semiconductor.

LOCK IT OR LOSE IT

A TimeKey contains missing but critical information necessary to make an applications software program work. Plugged into an IBM PC/XT/AT/PS2 printer port as part of a Dallas Semiconductor DS1255 KeyRing, the TimeKey unlocks the software. Users can still copy software for legitimate needs, but can't copy the TimeKey or its contents. Since programs won't run without the DS1207, unauthorized users are thwarted.

SUBSCRIPTION SOFTWARE

The DS1207 also allows publishers to set an expiration date on software, making it unusable after a prescribed time. This lets you offer potential customers a try-before-you-buy option, as well as leasing for customers who find lump-sum payments difficult for premium-priced software.

QUICK-INSTALLATION RECIPE

A "cookbook," the Dallas Semiconductor DS1255K Software authorization kit, brings you quickly up to speed on the TimeKey so you can put it to use immediately.

So stop being afraid of software pirates. Give yourself peace of mind and a revenue stream for the future with the DS1207 TimeKey from Dallas Semiconductor.

DALLAS SEMICONDUCTOR

4350 BELTWOOD • DALLAS, TEXAS 75244 • 214-450-0400

DISRIBUTORS: AVED, ADVENT, ALMAC, FUTURE, BELL, HALL-MARK, INDUSTRIAL COMPONENTS, INSIGHT, MILGRAY, QUALITY COMPONENTS, SRA, WESTERN MICRO, WYLE

CIRCLE 383 ON READER SERVICE CARD

(continued from page 20)

place.

Installing the Async AppleTalk Driver

Conventionally, a Macintosh application needing AppleTalk services checks to see if the AppleTalk driver is installed, and if it is, opens it. This offers us two options for installing Async AppleTalk on a disk: providing a way to open Async AppleTalk before an application checks to see if AppleTalk is open or replacing the standard AppleTalk driver so that an application will open ours instead. The simpler method is to replace the standard AppleTalk driver with our driver (in the system file). Then, when an application opens AppleTalk, Async AppleTalk comes up.

There are a few problems with using this scheme, however. Ordinary AppleTalk networks are permanently wired, whereas Async AppleTalk connections are usually temporary, over hard-wired or dial-up lines. The user may need to connect some cables, turn on a modem, or dial a phone to initiate a connection. In addition, Async AppleTalk must provide a way to reestablish a connection if a line is disconnected and to hang up the phone when finished. Finally, we wanted users to be able to use the same disk with standard AppleTalk and Async AppleTalk, and it isn't possible to have two AppleTalk drivers in the same system file.

Instead, we chose to write an Async AppleTalk installer as a desk accessory. Desk accessories are simple to write, and they can also run concurrently with applications. This lets users start up Async AppleTalk within MacWrite—for example, to access a LaserWriter—rather than having to remember to start it beforehand or forcing users to quit MacWrite, start up Async AppleTalk, and then reenter MacWrite.

The User Interface

A user runs the Async AppleTalk installer by selecting it from the Apple menu. The installer presents a window (see Figure 3, below) containing several controls and a status message display. First is a set of speeds, from 1,200 to 19,200 bits per second. To the right of this are four buttons that the user can click: Start, which loads and initializes Async AppleTalk (and can even dial the phone if necessary); Cancel, which terminates the desk accessory without doing anything; Hangup, which disconnects the phone and closes any AppleTalk driver; and Help, which describes what the installer is and how to use it.

Below those controls is a status area that displays the state of AppleTalk (installed, not installed, and so on) or messages from the installer. At the bottom of the window are controls for the auto-dialer. If the Auto-Dial check box is checked, the installer runs a program to dial the phone, using the phone number in the box, before making a connection. The user can enter a phone number before clicking the Start button.

A user starts up Async AppleTalk by selecting the installer from the Apple menu. When the dialog window appears, the user chooses a speed, checks Auto-Dial and enters a phone number if desired, and clicks the Start button. The installer then loads and opens the Async AppleTalk driver, dials the phone if requested, and negotiates for a network address. If all is well, the installer displays the message "Async AppleTalk Installed," saves the speed and phone settings, and terminates. If errors occur in loading the driver, the desk accessory displays a message and allows the user to correct the problem and try again. Common problems are loose cables, a modem that is turned off, an incorrect telephone number, or the other end is down. The user can exit from the desk accessory by clicking the Cancel button if the problem can't be corrected.

Because the installer desk accessory (DA) saves the settings of successful connections, the user usually only needs to select the desk accessory and click the Start button to start Async AppleTalk.

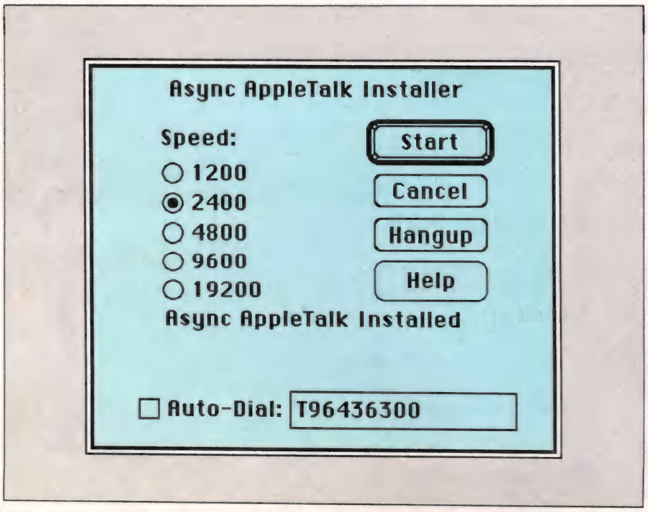


Figure 3: *The Async AppleTalk installer window*

Explanation of the Code

On the Macintosh, AppleTalk is implemented as a set of drivers. A driver is a group of routines that isolates an application from an I/O device by presenting a standard software interface. Because drivers aren't directly linked to an application, an alternative driver can easily be loaded in place of the standard code.

Each of the AppleTalk protocols is implemented as a group of subroutines (for ATP, NBP, DDP, and LAP). When an AppleTalk application requests an AppleTalk service, such as an ATP transaction, the ATP code prepares one or more messages. For each message to be sent, the ATP calls a routine (let's call it *DDPWrite*) to prepare a datagram. The *DDPWrite* routine finally calls a routine (*LAPWrite*) that actually sends the message out on the wire. Now the Async AppleTalk code comes into play. We took the source for standard AppleTalk and rewrote the *LAPWrite* routine to send the characters out on an async link instead of at 230.4 kbps. This produced a driver that was completely transparent to application programs. Our terminal emulation program, file sharing, printing, and other network services all operate unchanged over Async AppleTalk.

AALAP Environment

A low-memory variable (*AbusVars*) at address *$2D8* points to the local variables that Async AppleTalk needs. By convention, the AppleTalk code places this value in register

VAROOMM

ONLY ORCHID GIVES YOU SO MANY WAYS TO TURBOCHARGE PC ENGINES.

With Orchid's family of turbos, you can get the performance you bought a computer for in the first place. Lightning spreadsheet *recalc*ulations. CAD screens that *regen*erate in a flash. And large data bases that sort without putting you to sleep.

ORCHID'S TURBO FAMILY

	TinyTurbo 286	PCturbo 286e 10MHz	Jet 386
Host Computer	PC, XT	PC, XT, AT	AT
Reference Speed	3 x XT	6.5 x XT	3 x AT
Accelerator Type	Replaces 8088	Co-processor	Replaces 80286
CPU	8 MHz 80286	10 MHz 80286	16 MHz 80386
RAM	8 K Cache	1 or 2 MB EMS	64 K Cache

IF YOU HAVE AN	YOU GET	WITH
XT	3 X SPEED	TinyTurbo 286
XT	6.5 X SPEED	PCturbo 286e
AT	3 X SPEED	Jet 386

GET THE EDITOR'S CHOICE: TinyTurbo 286.™

The TinyTurbo 286™ supercharges your XT to run *three times faster.* Yet it's so small it takes up only half a slot in your computer. Just two reasons why *PC Magazine* named it the Editor's Choice.

TinyTurbo 286 gives you a high level of compatibility. So you can run software like Lotus and Windows—with EGA graphics, EMS memory, or networking cards—at AT speed. You can even go back to your PC's regular 8088 chip, which remains in the system, giving you 100% hardware compatibility.

ADD AWESOME PERFORMANCE: PCturbo 286e.™

For power users, the front runner today in accelerators is clearly the PCturbo 286e.™ It revs up to *6.5 times faster* than an XT, or up to *2 times* AT speed—giving you the world's fastest screen I/O. Plus the PCturbo 286e comes factory equipped with 1 Megabyte of fast RAM, expandable to 2.

The PCturbo 286e is also a powerful tool for developers and systems integrators. With features like an optional 10-MHz 80287 math chip, and coprocessing software for concurrent foreground/background tasks, the PCturbo 286e lets you build minicomputer-like performance into standard PCs.

MOVE YOUR AT UP TO 386 HORSEPOWER NOW: Jet 386.™

Take a look today at the price/performance leader in desktop computing: the Jet 386.™ Depending on the application, it's up to *three times faster* than an AT. And *twice as fast* as some high-performance minicomputers. Yet you don't have to buy an expensive 386-based computer to get this kind of horsepower.

More importantly, the Jet 386 uses next generation technology, the 80386 microprocessor. So you can run all of the current software for the AT now, and 386 software too. Add a Jet 386 to your AT today, and you can extend the life of your investment—for a fraction of the cost.

FROM THE COMPANY THAT STARTED IT ALL.

Orchid introduced the first turbo for PCs, and has since become the number one supplier of PC accelerators. For details on our full line of accelerators, graphics, networks and multi-function cards, call **(415) 683-0300** today. Or contact your local dealer.

TinyTurbo 286, PCturbo 286e and Jet 386 are trademarks of Orchid Technology. All other product names are trademarks of their manufacturers.

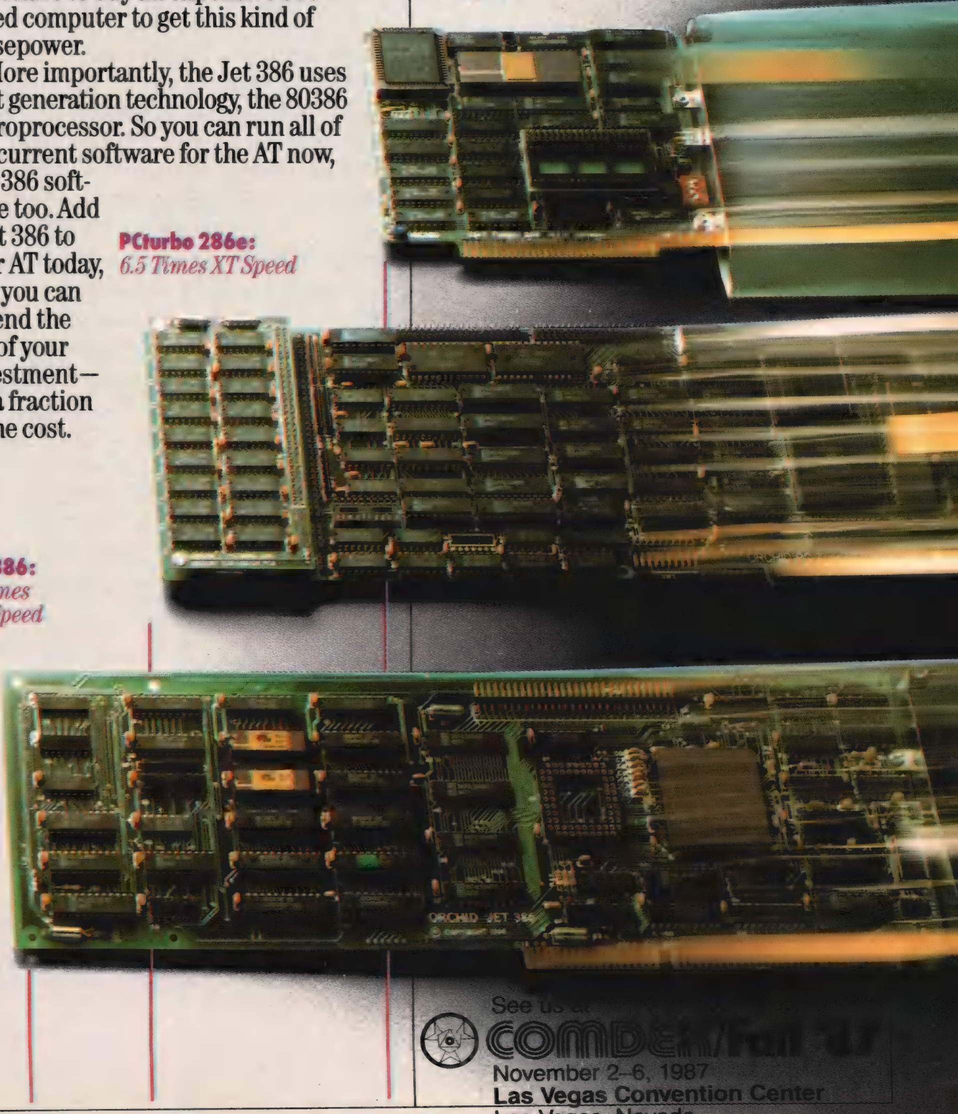
ORCHID TECHNOLOGY
45365 NORTHPORT LOOP WEST
FREMONT, CA 94538
(415) 683-0300; TLX 709289.

ORCHID

TinyTurbo 286: *3 Times XT Speed*

PCturbo 286e: *6.5 Times XT Speed*

Jet 386: *3 Times AT Speed*



3 x AT **6.5 x XT** **3 x XT**

See us at COMDEX/Fall '87
November 2–6, 1987
Las Vegas Convention Center
Las Vegas, Nevada
BOOTH no. 552

CIRCLE 130 ON READER SERVICE CARD

(continued from page 22)

A2 so that variables are at offsets from *A2*.

The Macintosh uses the (Zilog or AMD) 8530 Serial Communications Controller (SCC). This chip is capable of both async or synchronous transmission. Other serial chips (8250, 1602, and so on) also have the functions required for Async AppleTalk.

There are five important sections to the listings: sending a frame (*LAPWrite* and the transmit interrupt handler), receiving a frame (the receive interrupt handler), the CRC algorithm in 68000 assembly language and Pascal, the port A polling procedure, and miscellaneous support routines. (Listing One begins on page 60.)

Sending a Frame

Each time a frame is to be sent, the driver calls the *LAPWrite* routine to set up certain variables and send the first character of the frame. After that, the transmission is interrupt-driven—each time a character has been completely transmitted, the SCC interrupts the CPU. The interrupt handler sends another character and returns to the application.

LAPWrite is complicated by the fact that it must send frames "asynchronously"—this allows the initiator of the frame to continue without waiting for a long I/O operation, which is important because many seconds may elapse between the start and end of a frame. The device manager is responsible for queuing operations for drivers. It doles out tasks (say, to transmit a frame) one at a time. If an operation can be completed immediately, or if an error is detected, control returns immediately to the device manager.

When an asynchronous operation, such as transmitting a frame, cannot be completed immediately, control returns to the code that originally called the device manager. When the operation completes, the driver returns control to the device manager, which may initiate another operation. For more information about this scheme, refer to *Inside Macintosh, Volume II*.

LAPWrite uses a description of the frame called a write data structure (WDS). The WDS contains one or more segments of data to send as a single frame. Its format is a series of length-pointer pairs that describe each segment to be sent (see Figure 4, below). The length is a (16-bit) word, and the pointer is a (32-bit) pointer to the first character of the segment. Several length-pointer pairs can be concatenated in the WDS, with the final segment being followed by a word of 0.

Before sending a frame, *LAPWrite* first validates the frame by checking that the length is less than 600 data bytes.

Next, *LAPWrite* checks to see if there is currently a frame being transmitted. If so, it checks that it is an IM or UR frame (it returns an error otherwise), saves the WDS of the new frame in *qWDSptr* (the "queued WDS"), and returns (see accompanying article on page 25). If a frame is not being transmitted, *LAPWrite* updates the *PollProc* pointer (described later).

Finally, it ensures that the link is still up by checking that a valid frame has been received within 30 seconds. If not, *LAPWrite* sends an IM frame. A good response (a UR frame) updates the last-valid-frame timer, and the frame is sent as normal. The lack of a response after 10 seconds results in an alert to the user, who can use the desk accessory to restart the link.

Assuming the link is up, *LAPWrite* calls *SendWDSptr*, which saves a pointer to the current WDS in a variable *tWDSPtr* and then initializes several variables (*EscOut*, *nCRC*, *nFrmChr*) that will be used during the frame transmission. *SendWDSptr* also points the *LAPFetch* pointer at the first byte to send and sets *TxCount* to the length of the first segment. Finally, it sends the first character of the frame (*$A5*, the framing character) to the SCC. The transmit interrupt handler sends the remainder of the frame.

The transmit interrupt handler (*TIntHnd*) is an interrupt routine that is called each time the SCC finishes transmitting a character. This routine sends the next character (by calling *TxNextCh*) and cleans up before returning from the interrupt. If no character was actually sent (the previous character was the last one of the frame), then the transmit pending bit is reset so that no more interrupts will arrive. In any case, *TIntHnd* resets the highest interrupt under service (IUS) so that other interrupts can come in.

TxNextCh picks the next character of the frame to send, advances *LAPFetch* (the pointer to the next character to send), and decrements the segment length counter (*TxCount*). It also accumulates the CRC for the frame (by calling *NextCRC*). If escaping is necessary, *TxNextCh* sets the *EscOut* variable to true, leaves *LAPFetch* pointing at the character to be escaped, and sends a DLE (*$10*). If the *EscOut* flag is set, it exclusive-ORs the data character (at *LAPFetch*) with *$40*, increments *LAPFetch*, clears the *EscOut* flag, and sends the character. Notice that escaping always follows CRC accumulation on the transmit side. We will see that unescaping is done before CRC computations on the receive side.

There are several special cases for *TxNextCh*. When the last character of a segment of a WDS has been sent (*TxCount = 0*), it moves to the next length-pointer pair, updates *LAPFetch* and *TxCount*, and continues sending. After all the segments of a frame have been sent,

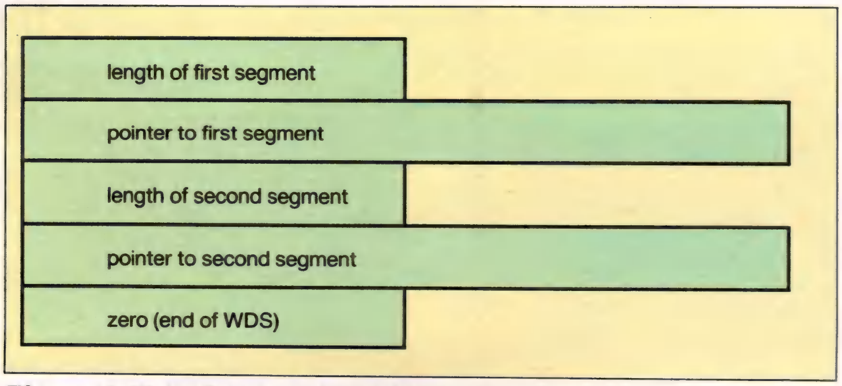


Figure 4: *Write data structure (WDS)*

The Async AppleTalk Link Access Protocol

This section describes the important features of AALAP. The full protocol definition, the Asynchronous AppleTalk Link Access Protocol, Version 1.0, is included on the Async AppleTalk distribution disk.

• AALAP transmits frames of arbitrary 8-bit data to another node across an RS-232 link. It uses 8-bit, no parity transmission with one stop bit. AALAP sends data bytes whose value is the same as certain special characters (that is, flow control characters) by escaping those characters; it sends a DLE (*$10*) and then the desired data character exclusive-ORed with *$40*. Thus the data byte *$13* (XOFF) becomes the two-character sequence *$10 $53*; to send a data byte of *$10* (DLE), AALAP sends the sequence *$10 $50*. The receiver always exclusive-ORs the character after a DLE with *$40* to recover the correct data byte. The characters treated specially are *$10*, *$11*, *$13*, *$91*, *$93*, and *$A5*.

• AALAP marks the beginning and end of each frame with a special "framing" character. This framing character never occurs in the middle of a frame; if AALAP needs to send it, it will be escaped. *$A5* is a good choice because it is unlikely to occur in most data streams and therefore will not have to be escaped often. Placing this character at the start and end of each frame allows easy resynchronization if one framing character has been garbled; the receiver simply waits for two adjacent framing characters, which indicate the start of another frame. Figure 5, below, shows the format of an AALAP frame.

• AALAP detects transmission errors using a cyclic redundancy check (CRC). AALAP computes the CRC (a 16-bit value that depends on all the bits of every character in the frame) and sends this value at the end of the frame. The receiver also computes the CRC on its received data and compares its computed value with the value it received. If the two values are different, an error occurred, and the receiver discards the frame. If the values are the same, then the frame is presumed to have been received correctly.

• AALAP links are full-duplex, point-to-point channels, so there is no contention for the link; standard ALAP has to arbitrate access to the bus to prevent two station's messages from colliding.

• AALAP negotiates the network and node number with the device at the other end of the link at start-up time. The AppleTalk address for a particular device consists of a (16-bit) network number and a (8-bit) node number (for the device itself).

• AALAP uses XON/XOFF flow control. This was a requirement for dealing with host computers that cannot accept large blocks of data at high speed. If AALAP receives an XOFF, it stops sending. It resumes output after receiving an XON. Similarly, AALAP may send an XOFF if it cannot process all the characters that have arrived. Some host computers send flow control characters of either parity (XOFF may be *$13* or *$93*) as flow control; AALAP must honor either.

• AALAP must detect and recover from link failures (such as loss of carrier). There are actually two problems: reestablishing the link and regaining the previous network address. On the Macintosh, a desk accessory (DA) accomplishes both. The DA will dial a selected telephone number, if necessary, and establish the link. If the link fails during a session, the user gets an alert and can then use the DA to restart the link.

• To start the link, the Async AppleTalk installer makes a link (by dialing the phone number and so on) and then gives a command to the AALAP driver to obtain a network and node number (NNN). This is a two-step process. Each end sends an IM (for "I aM") LAP frame that contains its suggested network address; the other end responds with a UR (for "yoU aRe") frame that either confirms that address or makes a different suggestion (see Figure 6, below). The default values for network and node numbers are *$0000* and *$00*, respectively. When zero values arrive in an IM frame, the recipient suggests different (possibly random) values in a UR response. If the recipient of a UR can accommodate the suggested value(s), it switches to use those numbers. When each end has sent an IM and received a UR with the same address, they declare the link to be up.

• Notice that this scheme allows a node to regain its previous network address after being disconnected. When restarting, the node suggests its previous address in the initial IM; if the other end can accommodate that address, it replies with the same values in the UR.

• Whenever a node wants to send a data frame, it checks to see if it has heard from the other end within the last 30 seconds. If not, it sends an IM frame with its current address (to force an immediate UR response). If no response returns, the link can be assumed to be down, and the user gets a warning. If a response arrives, then the original data frame can be sent.

Framing char	LAP type	up to 600 bytes of data	CRC	Framing char

Framing char — *$A5*
LAP type — *$02* DDP; *$86* IM; *$87* UR
Data bytes — arbitrary data
CRC — CRC-16 (see text)

Figure 5: *Generic AALAP frame*

Framing char	IM—*$86* UR—*$87*	Network number (2 bytes)	Node number (1 byte)	CRC	Framing char

Figure 6: *IM/UR frames*

(continued from page 24)

TxNextCh checks *nCRC* to see if it needs to send the CRC. If so, it swaps the bytes of the CRC (so that it will be sent the least significant byte first) and saves them in *CRCBuf*, points *LAPFetch* at *CRCBuf*, sets *TxCount* to 2, and clears the *nCRC* flag. Finally, after the CRC has been sent, if the *nFrmChr* flag is set, a closing framing character will be sent and the *nFrmChr* flag cleared. After the entire frame has been sent, *TxNextCh* looks to see if a queued WDS is waiting to be sent. If so, it resumes transmission with the new frame; otherwise, it informs the device manager that a frame is complete.

Receiving a Frame

The receive interrupt handler (*RIntHnd*) processes arriving characters. Every time a "read data available" interrupt occurs, *RIntHnd* gets a character from the SCC, processes the character as described later, and checks for another character. When there are no more characters, *RIntHnd* returns to the interrupted application.

The normal state of *RIntHnd* is waiting for a frame to begin. The variable *inMsg* is set false until a *$A5* character arrives. All other input characters will be discarded. Once a frame has started, each arriving character is checked for escaping. If it is a DLE, the *EscIn* flag is set and the DLE discarded. If a character arrives when the *EscIn* flag is set, the data character is exclusive-ORed with *$40* to get the correct value, and the *EscIn* flag is cleared. *RIntHnd* then updates the input CRC, stores the data character in the input buffer, and updates the pointers (*LAPStash* points at the next free location; *RcvdLen* is the total length of the input buffer).

When a closing framing character arrives, the *inMsg* variable is set false and the frame's received CRC is checked. A nonzero value means that a transmission error occurred, and the data is ignored. A CRC of zero implies that the frame is correct; *RIntHnd* remembers the frame's arrival time and passes control to a higher layer in the AppleTalk protocol for processing.

RIntHnd has several special tests. First, it discards frames that contain either fewer than 3 or more than 600 data characters (as required by the AppleTalk specification). *RIntHnd* also checks for flow control characters arriving from the other side and stops and resumes transmission as necessary. Finally, *RIntHnd* has some tricky code to keep up with arriving data while the processor is busy.

Remember that *RIntHnd* runs as an interrupt routine. No other interrupts will normally be processed until *RIntHnd* exits. Unfortunately, several of the higher protocol layers can take quite a while (3–4 msec) to process a long frame. At 9,600 bps, interrupts are disabled for three or four character times and characters are frequently overrun, corrupting the frame.

To avoid dropping characters while a frame is being processed by a higher layer, *RIntHnd* uses the *stillBusy* flag to indicate that the previous frame is still being processed. When a complete frame arrives, *RIntHnd* sets *stillBusy* and reenables interrupts before passing control to the higher layer's code. Characters that arrive when *stillBusy* is true are placed in a buffer (*BusyBuff*) of 16 characters. Once the higher layer returns, *RIntHnd* clears *stillBusy* and exits.

There is one final complication in these routines—the Macintosh doesn't have enough speed to service the disk and its two serial ports simultaneously. This means that incoming characters may be dropped when the disk is spinning. The disk driver installs a "serial port A polling procedure," or *PollProc*, which is invoked whenever the disk driver turns off interrupts for more than 100 microseconds. During disk operations, *PollProc* stashes characters from port A into a special buffer for processing after the disk stops.

Unfortunately, the Async AppleTalk driver (on port B) drops characters with this scheme. To circumvent this, we install a small piece of code to be executed before the standard *PollProc* runs. This code has one function: to send an XOFF flow control character to stop the other end from sending. A while later, AALAP sends an XON to resume the data (the vertical blanking task controls this).

CRC Algorithm

The two CRC routines shown in Listings Two and Three, next month, implement the same algorithm. Listing Two is in Pascal and is easier to follow than the actual assembly-language code used (in Listing Three). Each *NextCRC* routine receives the accumulated CRC (so far) and the next data character to accumulate. *NextCRC* returns the updated (16-bit) CRC. The CRC-16 algorithm has the special property that when the computed CRC is appended (least significant byte first) to a string of m data bytes, and a CRC computed on the entire m+2 bytes, the result is exactly 0. This makes it quite easy for the receiver of a frame to check the CRC of a frame.

The Polling Procedure (PollProc)

Our *PollProc* gets control when the disk driver turns off interrupts. It first tests to see if first, it has sent an XOFF, and second, if we're receiving a frame. If the answers are no and yes, respectively, it stashes any characters from the SCC in *BusyBuff* and then sends an XOFF to the other end.

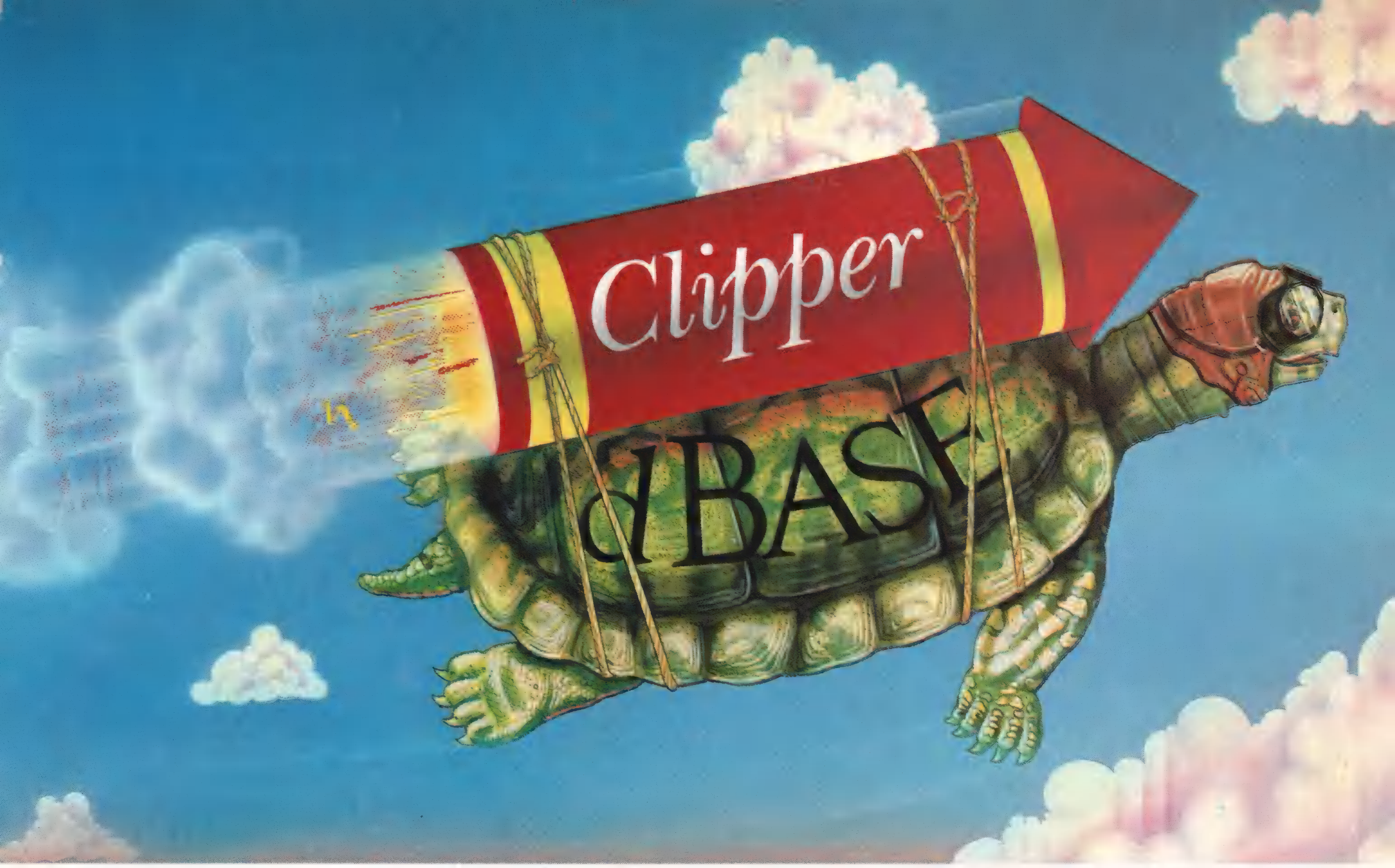
In any case, our *PollProc* then restores the state and passes control to the real *PollProc* (if any).

The AALAP *PollProc* only sends an XOFF when actually receiving a frame (when *inMsg* is true). This prevents a continuous dribble of XOFF and XON characters as the disk spins.

Support Routines

Macintosh programs can receive periodic action with a vertical blanking (VBL) task. Async AppleTalk uses a VBL task to check that the data flow hasn't stopped without reason. The VBL code runs every third of a second and either sends an XON if the transmitter has sent an XOFF to stop the other side or experimentally sends the next character (by calling *TxNextCh*) if no character has been sent for a full second. Each of these actions may reinvoke flow control, but they guarantee that things will become "unstuck" if, say, an XON was dropped.

PutChar sends a character synchronously—that is, it waits until the character can be sent. If the SCC cannot accept the character within one-half second, *PutChar* re-



Turtle Souped

Real programmers don't use dBASE. Or do they?

We're finding that some very swift programmers are using it to write some very fast applications, and are completing their projects much more quickly.

But they cheat.

They use our Clipper™ compiler to combine dBASE™ with C and assembler.

With dBASE used like pseudo-code, they can then quickly create prototypes that actually run.

Then, with dBASE doing the high-level database functions, they use our Clipper compiler to link in C or assembly language modules from their own bag of tricks.

And they're finding that they're linking in less than they expected because Clipper compiled code runs so fast and because of Clipper's built-in enhancements.

Clipper includes easy networking that provides file and record locking the way it should be done.

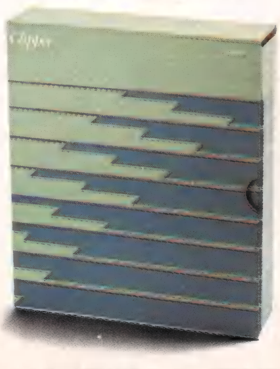
Fast screens that can be treated as memory variables and eliminate the need for direct screen writes and all that tortuous heap management code.

Box commands that make windowing a breeze. And more.

So if you'd like to use your time more productively, check us out: Nantucket Corporation, 12555 W. Jefferson Boulevard, Los Angeles, CA 90066.

Or if you're on deadline, call (213) 390-7923 today.

Clipper could get you out of the soup.



Nantucket®

© Nantucket Corporation 1987. Clipper is a trademark of Nantucket Corporation; dBASE isn't. In Europe: Nantucket Corporation (Europe) 2 Bluecoats Avenue, Fore Street, Hertford, Herts SG14 1PB Telephone 0992 554621.

CIRCLE 220 ON READER SERVICE CARD

(continued from page 26)

turns an error code. A character of *$FFFF* will send a break on the link.

SetBaud changes the baud rate generator (BRG) in the SCC to run at the desired speed. It is careful to disable the BRG before changing it and restores the state of the SCC before returning.

Get_NNNN does the network and node number negotiation (NNNN). It sends an IM with the current network address up to four times. A UR response will be processed asynchronously by the receive interrupt handler. If the address in the UR matches that of the IM, then *Get_NNNN* declares the link up. If the *AALAPup* variable is true before the fourth IM time-out, *Get_NNNN* returns a good status; otherwise, it returns a bad status to indicate that the link could not be started.

DoWarn uses the return status from *Get_NNNN* to display a dialog box with a warning to the user that there is trouble with the Async AppleTalk link. Two possibilities are likely: either there is no response from the other end or the two ends of the link cannot agree on a network address. *DoWarn* will fail if the disk that originally contained the Async AppleTalk desk accessory file is not currently mounted.

Who's Using Async AppleTalk?

At Dartmouth, more than 1,000 Macintoshes in administrative and academic offices can use Async AppleTalk through the Kiewit network. We have successfully used many applications, including DarTerminal, printing to LaserWriters, and various file servers and electronic-mail programs, with Async AppleTalk.

At the time of this writing (May 1987), Async AppleTalk is only useful on the Dartmouth network. Fortunately, commercial products that use Async AppleTalk should soon reach the market. First is the Reactor, from Sand Hill Engineering in Geneva, Florida ([305] 349-5960). The Reactor is a port switcher that can interconnect RS-232 ports internally or between several Reactors, which allows multiple users to share a modem, printer, or whatever, without switching cables. The Reactor also acts as an Async AppleTalk bridge, routing AALAP frames between the appropriate ports.

Solana Electronics, of San Diego, California ([619] 566-1701), is also working on an Async AppleTalk product that will connect to a 230.4-kbps bus.

Future Projects

Although the current implementation of Async AppleTalk is quite usable, several enhancements could be made. Perhaps the simplest would be to make an INIT resource that installed Async AppleTalk at system boot time. Each time you booted the system, the INIT mechanism of the Mac would load in the Async AppleTalk driver, dial the telephone (if necessary), and establish the link. Errors would be reported in alerts as they are currently.

Another project is a "half-bridge." This is most easily described as a stand-alone Macintosh that has Async AppleTalk running over the phone port (port A) and standard (230.4-kbps) AppleTalk running on the printer port (port B). Frames that arrived on port A would be sent out port B, and vice versa. This is a bit tricky because not all frames need to be sent out the other port—the software in the half-bridge should forward frames only if the destination is out the other side. This would be useful for a couple of purposes: a half-bridge can join two separate 230.4-kbps AppleTalk networks, only passing frames destined for the other side; a half-bridge also gives a dial-up port for someone using Async AppleTalk, say, from home. (This isn't as wasteful as it seems: it is quite reasonable to leave a Mac in the office running as a half-bridge after people leave. After all, many people turn the Mac off when they're not there.)

Availability

Although you may distribute it freely, Async AppleTalk is not in the public domain. It bears a copyright notice of the Trustees of Dartmouth College. We distribute it at a nominal cost, and others may redistribute it as long as it is not sold for profit and as long as the copyright notice is maintained. The full distribution policy is included with the distribution disk. If you plan to use Async AppleTalk in a commercial product, we ask that you send us a letter describing your plans and your agreement to abide by our policy. (Historical note: this is fundamentally the same arrangement Columbia University uses for its Kermit package.)

Dartmouth College distributes the following modules:

- A desk accessory for the Macintosh.
- Sources for the desk accessory and auto-dialer.
- Sources for most, but not all, of the AALAP driver. AALAP was developed from the original AppleTalk source code, which we licensed from Apple under a nondisclosure agreement. Consequently, we cannot distribute the entire source package. We do show most of the software, including interrupt handlers, so that these can be ported to other machines.

All source code for articles in this issue is available on a single disk. To order, send $14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94036, or call (415) 366-3600, ext. 216. Please specify issue number and format (MS-DOS, Macintosh, Kaypro).

Bibliography

Apple Computer. *Inside AppleTalk*. Cupertino, Calif.: Apple Computer, 1986.

Apple Computer. *Inside Macintosh*. vols. I–IV. Reading, Mass.: Addison-Wesley, 1985.

Dartmouth College. *Asynchronous AppleTalk Link Access Protocol, Version 1.0*. Hanover, N.H.: Kiewit Computation Center, 1986.

Tanenbaum, Andrew S. *Computer Networks*. Englewood Cliffs, N.J.: Prentice-Hall, 1981.

DDJ

(Listing begins on page 60.)

Vote for your favorite feature/article.
Circle Reader Service **No. 1**.

Clarify and document your source listing and get an "organization chart" of your program's structure

with two NEW utilities from Aldebaran Laboratories, for C, BASIC, Pascal, dBASE,® FORTRAN and Modula-2 programmers.

Now works with FORTRAN

"Occasionally, a utility comes along that makes a programmer's life much easier. SOURCE PRINT is such a program. It contributes to the programmer's job by organizing code into a legible format and by helping to organize the documentation and debugging process."

— PC Magazine Sept. 16, 1986

Source Print and Tree Diagrammer both have easy-to-use menus with point-and-shoot file selection, and let you search for files containing a given string. For IBM PC and compatibles with 256K.

Join thousands of programmers who are working more efficiently using Source Print and Tree Diagrammer. Order these indispensable tools today. We ship immediately, and there's no risk with our 60-day money-back guarantee. **Order both and save. Only $155.00.**

800-257-5773 Dept. 58
In California:
800-257-5774 Dept. 58

MasterCard, VISA, American Express, COD. Add $5 for shipping/handling.

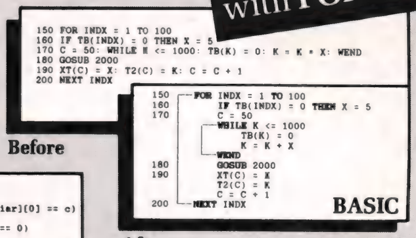
or see your local dealer!

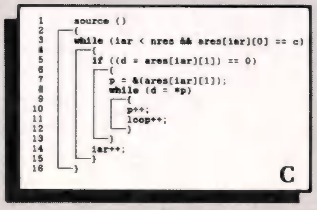
Source Print and Tree Diagrammer are trademarks of Aldebaran Labs. dBASE is a trademark of Ashton Tate. Prices subject to change without notice.

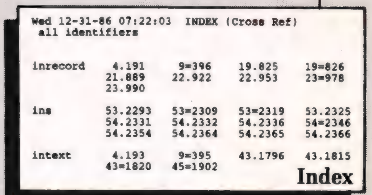
Source Print™

organizes your source code, simplifies debugging, and makes documentation a snap! It lists one or more source files with informative page headings and optional line numbers, while offering invaluable features:

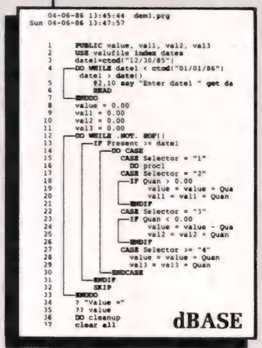
The Index (Cross-Reference list) saves you time by showing exactly where variables are used and where functions, procedures, and routines are called. Locations where new values may be assigned to variables are shown, making it easy to track down that mysterious value change.







$97.00



Structure Outlining solves the problem of hard-to-see nested control structures by automatically drawing lines around them.

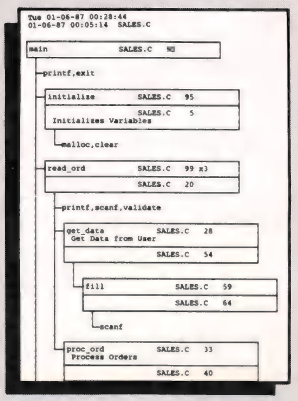
Automatic Indentation of source code and listings reduces your editing time and ensures indentation accuracy.

Plus . . . Source Print generates a table of contents listing functions and procedures. Keywords can be printed in boldface on most printers. Multi-statement BASIC lines can be split for readability. Functions and procedures can be drawn by name from one or more source files to form a new file.

Tree Diagrammer™

shows your program's overall organization at a glance. Ordinary program listings merely display functions, procedures, and subroutines sequentially, but do not display the relationships between these routines. Our revolutionary new Tree Diagrammer automatically creates an "organization chart" of your program showing the hierarchy of calls to functions, procedures, and subroutines. Recursive calls are indicated and designated comments in the source code will appear on the chart.

Tree Diagrammer helps you organize your program more logically. And you'll be amazed at how easy it is to debug when you see how your routines interact.



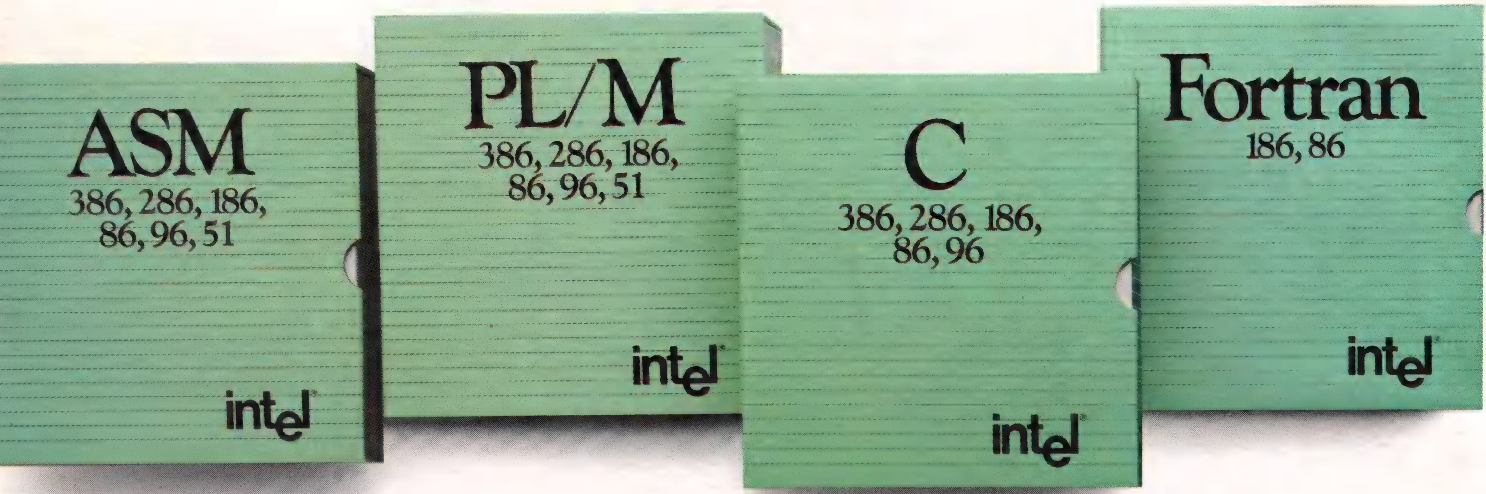
$77.00

Aldebaran Laboratories 3339 Vincent Rd. Pleasant Hill, CA 94523 415-930-8966

YES! Rush me ☐ Source Print @ $97. ______ ☐ Tree Diagrammer @ $77. ______
☐ Both $155. Ship/Handling $5. For CA add 6% tax ______ Total ______
Name ______
Company ______
Address ______
City ______ State ______ Zip ______
☐ Check enclosed ☐ VISA ☐ MasterCard ☐ American Express
Card # ______ Exp. Date ______
Signature ______ Phone # ______ 58

CIRCLE 350 ON READER SERVICE CARD

WRITE FASTER IN ANY LANGUAGE.



If you develop software for any product based on an Intel microcontroller or microprocessor, including the 80386, the unique debug hooks in the Intel languages will help get the job done faster.

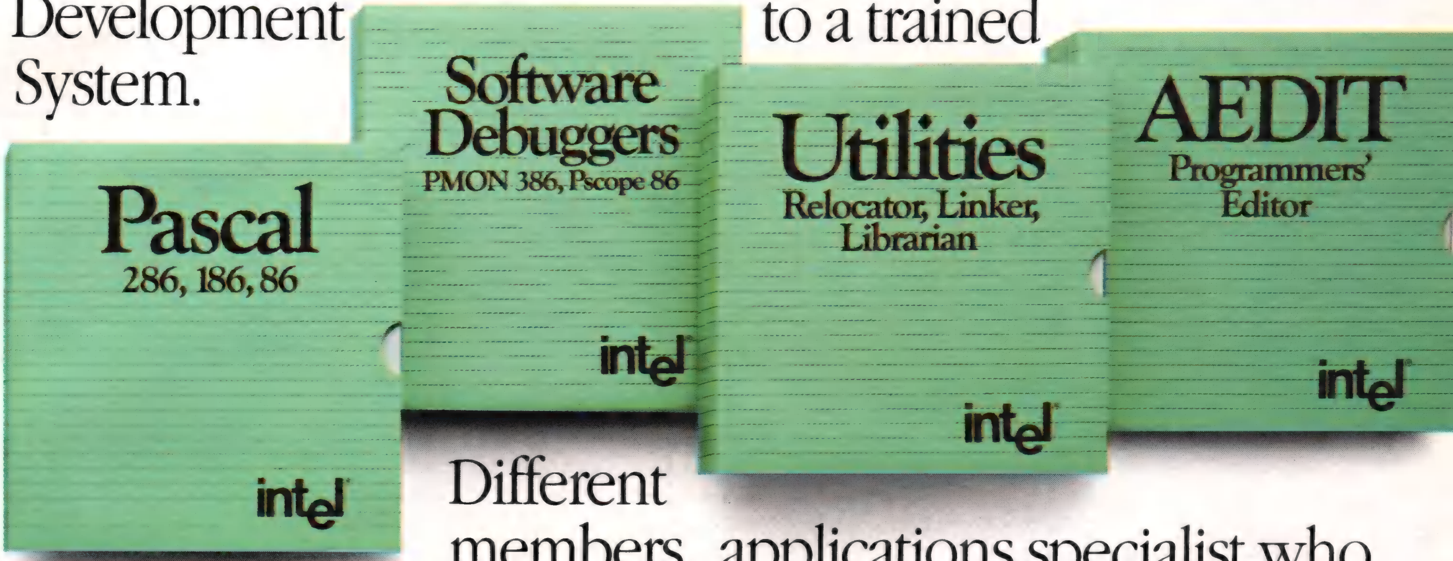
In fact, when used with Intel debuggers and emulators, Intel development languages can provide more debug data than any other high-level language.

Debug hooks let you symbolically debug in the same high-level language you wrote in without having to deal with machine or hex code. Which means 80.386 reads as 80.386, not 50 62 D0 C5.

Because the location of both code and data are easily specified with our locator, it is easier for you to develop ROM-based firmware.

Since Intel languages

produce identical object code regardless of the host, you can write code at a PC running DOS, a VAX*/VMS terminal, or an Intel Development System.



Different members of the same design team can therefore choose the most effective combination of languages and systems to get the job done faster.

Intel post-sales support can also help you get the job done faster. We invented the microprocessor. We know microprocessors and languages for Intel architectures better than anyone else.

When you buy an Intel language, you have access to our customer hotline. So if you ever have a question you can talk directly to a trained applications specialist who understands our products. And can give you the right answers. Faster.

To order today, or get more information—including a free catalog of our development tools—call toll-free 1-800-87-INTEL.

The sooner you call, the faster you'll get the job done.

intel®

© 1986 Intel Corporation
*VAX is a registered trademark of Digital Equipment Corporation.

CIRCLE 179 ON READER SERVICE CARD

A Fast Forth for the 68000

Saying good-bye to the inner interpreter

by Lori Chavez

Interactive languages such as Forth and BASIC traditionally force programmers to sacrifice program execution speed for the powerful debugging facilities of an interactive programming environment. This article takes a look at a Forth implementation scheme that allows an interactive Forth system to generate code that executes at speeds that rival the execution speeds of code generated using compiled languages.

Mach2, created for the Macintosh and other 68000 environments, is a Forth system that has discarded the elaborate pointer-threading schemes commonly used for Forth in favor of a simple "subroutine-threaded" approach. In a subroutine-threaded Forth, the Forth compiler generates machine code. Both the pointer-threaded Forth pseudocode and the inner interpreter required to execute the pseudocode are eliminated; the microprocessor can directly execute subroutine-threaded Forth code.

In benchmark tests performed on a Macintosh SE, the Sieve (Forth version) benchmark compiled by Mach2 executed at 70 percent of the execution speed of a Sieve program compiled using a high-performance Macintosh C compiler. The Sieve (Forth version) benchmark compiled using a traditional pointer-threaded Forth system executed at 17 percent of the speed of the compiled C version.

A Short Description of Forth

The Forth language is composed of approximately 200–300 routines. In Forth terminology these routines are referred to as words, or definitions. The data structure used to hold the code for the routines, information about the routines, and to link all the routines is called the dictionary. Each Forth word has its own entry in the dictionary.

New Forth definitions are created by compiling references to several previously compiled Forth words and assigning a name to those compiled references. The new Forth definition, like any other Forth word, can then be executed interactively by the Forth interpreter. Execution of the new definition will cause all words referenced by the definition to be executed sequentially.

To illustrate this process, let's define a Forth word called *SHIP* that, when executed, will perform the steps required to ship a product from a warehouse:

```
: SHIP
GET_QUANTITY
PREPARE_SHIPMENT
SEND_PACKAGE
;
```

GET_QUANTITY fetches the desired quantity value from a variable and passes the value to *PREPARE_SHIPMENT*. *PREPARE_SHIPMENT* places the desired number of items in the package and *SEND_PACKAGE* puts the package in the mail.

Subroutine vs. Pointer Threading

Figure 1, page 33, shows the code generated when *SHIP* is compiled in a subroutine-threaded system. Each word that *SHIP* references is a subroutine that ends in an assembly-language return-to-subroutine (RTS) instruction. The compiler generates a 4-byte, PC-relative, jump-to-subroutine (JSR) reference for each of the three routines. Because *SHIP* itself ends with an RTS instruction, it too can be referenced by another word in the same manner.

Figure 2, page 33, shows how the dictionary entry for *SHIP* would appear if it were compiled in a pointer-threaded Forth implementation. A pointer-threaded Forth does not generate directly executable machine code. Instead, it generates lists of either addresses, offsets, or tokens that indirectly "point" to the referenced word. This means a Forth interpreter that understands this "pseudocode" generated by the pointer-threaded compiler must be used to execute the "code."

Why Pseudocode?

Forth was originally developed in the days of the 8- and 16-bit processors. Typically, only one of the registers in these processors—the system stack pointer register—could take advantage of the fast processor stack manipulation operators (push and pop).

In a pointer-threaded Forth, the processor system stack is used as the data stack. Because microprocessors use the system stack for their subroutine calling mechanism, pointer-

Lori Chavez, Palo Alto Shipping Company, P.O. Box 7430, Menlo Park, CA 94026. Lori is a codeveloper of the Mach2 Forth Development System and is also a software consultant.

threaded Forth implementations were forced to develop their own definition execution mechanism. The scheme devised involved a simulation of the microprocessor program counter, instruction pointer, and even the instruction set. For those interested, Ronald Greene provides a listing of the 8088 assembly-language code required to implement a generic inner interpreter in his 1984 *Byte* article on reducing overhead in threaded interpretive languages. The inner interpreter is naturally slower than the microprocessor's subroutine calling mechanism and requires more registers to implement.

It is this simulation of the natural functions of the microprocessor that became the bottleneck in the effort to achieve fast Forth execution times. At the time, given the architecture of the available processors, the Forth implementors' decision made sense. The data stack is heavily used in a Forth program. By using the system stack as the data stack, fast push and pop stack manipulation instructions could be used to optimize parameter passing. And, in those days of limited memory, the use of subroutine call instructions would have required more memory for each compiled reference in a Forth word.

Today, however, processors are much more flexible. The Motorola 68000 microprocessor has 16 general-purpose registers (*D0–D7* and *A0–A7*), 7 of which (*A0–A7*) can be used as stack pointer registers. This means that a 68000 Forth can take advantage of the 68000's subroutine calling mechanism, which affects the system stack pointer register, and still use another register for the parameter stack pointer with no speed penalty.

Optimization Techniques

As Figure 1 demonstrated, a subroutine-threaded Forth compiler generates machine code. Although this may seem only natural to those familiar with other language compilers and assemblers, it is not a common characteristic of Forth. This change from pseudocode to machine-code generation has had two positive effects on the Forth language. First, as the previously mentioned Sieve results testify, subroutine threading makes Forth code execute significantly faster. Second, the code and speed optimization techniques traditional compilers and assemblers have used for many years can finally be applied to Forth code.

The rest of this article discusses how macro substitution—a speed improvement technique commonly used by assemblers—and edge-macro (peephole) optimization—a code optimization technique commonly used by traditional language compilers—can be used by a subroutine-threaded Forth compiler to generate faster and more compact code.

Optimization for Speed

To obtain speed improvements in an assembly-language program, programmers generally avoid subroutine calls and place as much code as possible directly into the routine being optimized. If this technique is used too often in a program, the source listing can become quite long and unreadable. Therefore, most assemblers include some sort of macro facility that allows many instructions to be represented by one word. When the macro word is encountered during the assembly process, all instructions that comprise the macro word will be assembled.

"Mach" words are the Mach2 equivalent of a traditional assembler's macro words. In Mach2, one bit in the length byte of the dictionary header is used as the Mach bit. If this bit is set, the corresponding word is a Mach word that is treated as a macro by the compiler. Whenever a Mach word is referenced in a definition, the compiler lays a copy of all instructions from the start of the Mach word up to the first RTS encountered into the definition being compiled. Because the compiler in a subroutine-threaded Forth already generates machine code, teaching it to produce in-line code is a simple task.

To decide which Forth words should be laid in-line, the overhead of the subroutine call instruction must be weighed against the size of the subroutine being called. The 68000 PC-relative JSR instruction requires 4 bytes of memory and 18 processor clock cycles. The 68000 RTS instruction requires 2 bytes of memory and 16 processor clock cycles. The total overhead for a complete subroutine call, for both the JSR and RTS instructions, is therefore 6 bytes and 34 clock cycles. For approximately 75 percent of the words in the Forth kernel, the overhead of the subroutine calling mechanism is small in comparison to the size and time required to execute the word itself. For the remaining 25 percent of the words, however, the overhead involved in calling the words is much greater than the execution time and memory requirements of the word being called.

An ideal candidate for a Mach word is a word such as *DUP*, which is composed of a single 68000 instruction (*MOVE.L (A6),–(A6)*) that is both smaller in size than the JSR instruction (2 bytes vs. 4 bytes) and faster in

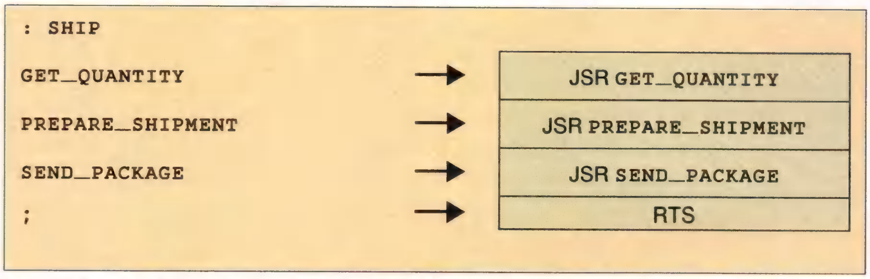


Figure 1: *Subroutine-threaded code for* SHIP

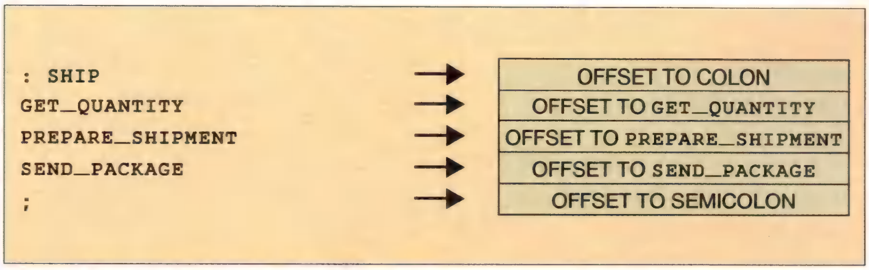


Figure 2: *Pointer-threaded code for* SHIP

68000 FORTH
(continued from page 33)

execution time than a subroutine reference (20 processor cycles for *DUP* vs. 54 processor cycles for *JSR + DUP + RTS*).

In Mach2, all of the simple stack, arithmetic, comparison, and memory operators and all variable and constant references have been declared as Mach words. Fifty percent of these words generate code that is smaller than or equal in size to the JSR instruction they replace. An additional 25 percent of the words weigh in at 6 bytes—only 2 bytes larger than the JSR instruction. In each case, the replacement of the JSR instruction with in-line code results in at least a 50 percent speed improvement. When used thoughtfully and selectively, macro substition can yield a large increase in program execution speed with only a small increase in program size.

One example, the definition of *GET_QUANTITY*, is shown below. *QUANTITY* is the name of a 4-byte variable storage area. When executed it will return the address of its storage area. The @ operator (fetch) is used to fetch the 4-byte contents of the *QUANTITY* variable.

```
: GET_QUANTITY   ( – n ) \ n means
     a value is returned on the stack.
  QUANTITY  \  Return variable
               address
  @ ;          \ Fetch 4-byte value
               from variable
```

Both the variable reference and @ are Mach words. By placing *QUANTITY* and @ directly into *SHIP*, you can eliminate the overhead of the *GET_QUANTITY* subroutine call and experience the benefits of macro substitution.

Figure 3, below, shows the code generated for *SHIP* when the code for the *QUANTITY* variable reference and the @ memory access are treated as assembly-language macros and laid in-line into *SHIP*'s code area. As you study Figure 3 note that Mach2 variables are located relative to the address in the *A5* register and that the *A6* register is used to maintain its parameter stack.

With macro substitution, the 4-byte JSR *QUANTITY* is replaced with the 4 bytes of assembly-language code that perform the variable reference. Likewise, the 4-byte JSR @ is replaced with the 4 bytes of assembly-language code that perform the memory access. *SHIP* doesn't increase in size but the *QUANTITY @* part of *SHIP* experiences a 57 percent speed improvement.

Code Optimization

Because Forth words use a stack for parameter passing, it is common for the first instruction in a Forth word to pull a parameter from the stack (*MOVE.L (A6)+,A0*) and for the last instruction to put a parameter on the stack (*MOVE.L A0,–(A6)*). When two Forth Mach words that have these common beginning and ending instructions are butted together, as in Figure 3, the result is the following redundant and inefficient code sequence:

```
. . .
MOVE.L A0,–(A6)
MOVE.L (A6)+,A0
. . .
```

Fortunately, it is not hard for the compiler to watch for these edge macro conditions and eliminate unnecessary code if possible (see Figure 4, below). With the use of this edge macro optimization, the *SHIP* code becomes faster and more compact. Two instructions, 4 bytes, can be removed from the *QUANTITY @* code and the required clock cycles then drop from 52 cycles to 28 cycles.

Conclusions

The advantages of subroutine threading are many—for example:

1. Speed—By removing the inner interpreter and using code and speed optimization techniques, subroutine-threaded Forth code executes three to four times faster than pointer-threaded Forth code.
2. Sharable code—Linking to other languages is simpler. Because all languages involved use only machine

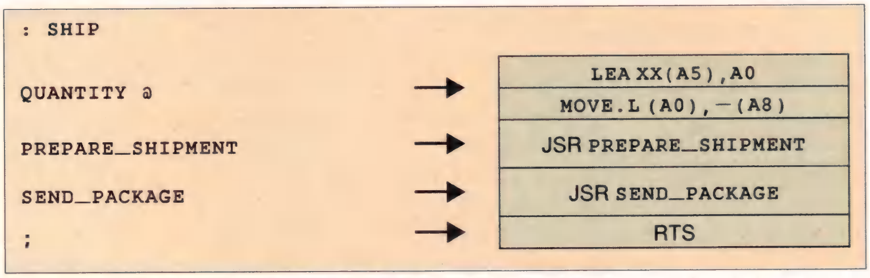


Figure 4: *Subroutine-threaded code for* SHIP *(with macro substitution and edge macro optimization)*

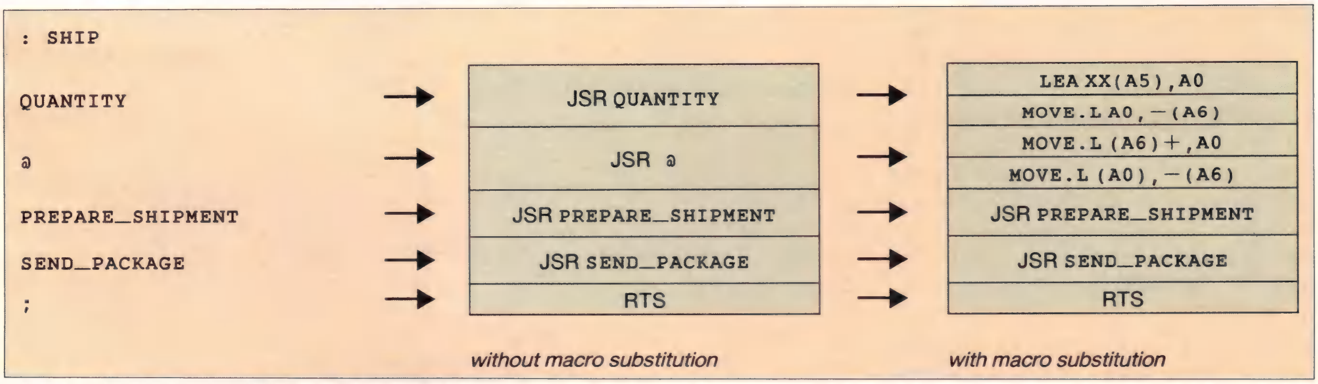
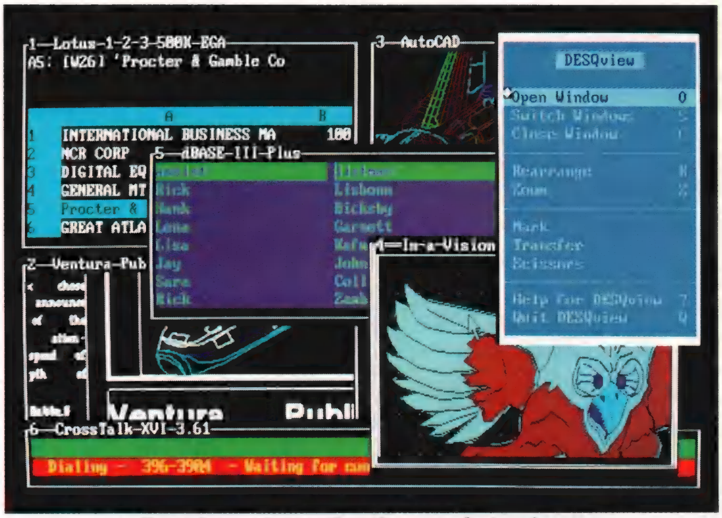


Figure 3: *Subroutine-threaded code for* SHIP

How to tell the difference between DESQview™ 2.0 and any other environment.

Selecting DESQview, the environment of choice, can give you the productivity and power you crave, without the loss of your old programs and hardware. If you like your existing programs, want to use them together, transfer data between them, print, sort, communicate with or process-in-background, yet still have the need to keep in place your favorite PC(8088, 8086, 80286 or 80386), DESQview is the "proven true" multitasking, multiwindowing environment for you. Best of all, DESQview 2.0 is here now, with all the money saving, time saving, and productivity features that others can only promise for the all-too-distant future.



One picture is worth a thousand promises.

And with DESQview's new graphics enhancements for Hercules, CGA, EGA, and VGA, Version 2.0 still offers the same award winning and pioneering features for programs that earned DESQview its leadership, only now you can also run desktop publishing programs, CAD programs, even GEM-™, Topview-™, and Microsoft Windows-™ specific programs. In some cases you'll add as little as 10–40K to your system overhead. Now you can have multi-tasking, multi-windowing, break the 640K habit too and still get an auto dialer, macros, menus for DOS and, for advanced users, a new complete application programmer's interface capability. No wonder that over the years, and especially in recent months, DESQview, and now DESQview 2.0 have earned extravagant praise from some of the most respected magazines in the industry.

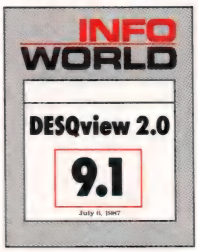
"Product of the Year" by readers vote in InfoWorld.

"Best PC Environment" by popular vote at Comdex Fall in PC Tech Journal's "System Builder" Contest.

"—I wouldn't want to run an IBM or compatible computer without DESQview"—InfoWorld, Michael Miller.

"A colossus among windowing environments"... "will run almost anything"—PC Week, Marvin Bryan.

"Windows, promises, but DESQview delivers"—MICROTIMES, Birell Walsh.



No other environment has consistently pioneered features, openness, and productivity. See for yourself. Send in the coupon. The possibilities are endless with DESQview 2.0.

Attention Programmers: For more information about Quarterdeck's API, and future 386 program extensions, call us today.

SYSTEM REQUIREMENTS
IBM Personal Computer and 100% compatibles (with 8086, 8088, 80286 or 80386 processors) with monochrome or color display; IBM Personal System/2 ▪ Memory: 640K recommended; for DESQview itself 0-145K ▪ Expanded Memory (Optional): expanded memory boards compatible with the Intel AboveBoard; enhanced expanded memory boards compatible with the AST RAMpage ▪ Disk: Two diskette drives or one diskette drive and a hard disk ▪ Graphics Card (Optional): Hercules, IBM Color/Graphics (CGA), IBM Enhanced Graphics (EGA), IBM Personal System/2 Advanced Graphics (VGA) ▪ Mouse (Optional): Mouse Systems, Microsoft and compatibles ▪ Modem for Auto-Dialer (Optional): Hayes or Compatible ▪ Operating System: PC-DOS 2.0-3.3; MS-DOS2.0-3.2 • Software: Most PC-DOS and MS-DOS application programs; programs specific to TopView 1.1, GEM 1.1 and Microsoft Windows 1.03 ▪ Media: DESQview 2.0 is available on either 5¼" or 3½" floppy diskettes

Rush me DESQview 2.0! Today! *DDJ 10/87*

No. of Copies	Media 3½"/5¼"	Product		Retail Price ea.	Total
		DESQview 2.0		$129.95	$
	Shipping & Handling		USA Outside USA	$ 5.00 $ 10.00	$
		Sales Tax (CA residents)		6.5%	$
				Amount Enclosed	$

Payment: ☐ Visa ☐ MC ☐ AMEX ☐ Check

Credit Card: Valid Since ______/______ Expiration ______/______

Card Number: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Credit Card Name ______

Shipping Address ______

City ______ State ______ Zip ______ Telephone ______

Mail to: Quarterdeck Office Systems, 150 Pico Boulevard, Santa Monica, CA 90405.
NOTE: If you own DESQview call us for a special upgrade offer, or send in your DESQview registration card. AST Special Edition users included.

DESQ view™
Quarterdeck

See us at COMDEX®/Fall '87
November 2–6, 1987
Las Vegas Hilton Hotel
Las Vegas, Nevada

Quarterdeck Office Systems • 150 Pico Boulevard, Santa Monica, CA 90405 • (213) 392-9851

DESQview is a trademark of Quarterdeck Office Systems. AboveBoard is a trademark of Intel Corporation. Hayes is a trademark of Hayes MicroComputer Products Inc. IBM, PC, Personal System/2 and TopView are trademarks of International Business Machines Corporation. Microsoft Windows and MS are registered trademarks of Microsoft Corporation. Mouse Systems is a trademark of Metagraphics/Mouse Systems. RAMpage is a trademark of AST Research, Inc. GEM is a trademark of Digital Research. Hercules is a trademark of Hercules.

CIRCLE 234 ON READER SERVICE CARD

68000 FORTH
(continued from page 34)

code instructions, only register saving and setup are required.

3. Conventional—The implementation is easy for non-Forth programmers to understand and use.
4. Relocatable code—The use of PC-relative subroutine references means the code generated is relocatable.
5. Well-suited to systems-level programming—It is easy for a subroutine-threaded compiler to generate stand-alone assembly-language routines, such as those required for device drivers.

Subroutine threading has disadvantages in the areas of size of generated code and the ease of disassembly.

Regarding size, with the inclusion of a subroutine call instruction with each reference and the use of macro substitution, the code generated by a subroutine-threaded Forth will be 50–100 percent larger than code generated by a pointer-threaded Forth.

The ease of disassembly is, happily, not a problem every software developer will face. Subroutine-threaded code is easier to disassemble than pointer-threaded code. Therefore it is correspondingly harder to disguise proprietary algorithms written in a subroutine-threaded Forth.

Bibliography

Greene, Ronald. "Faster Forth: Reducing Overhead in Threaded Interpretive Languages." *Byte* (June 1984): 127.

Loeliger, R. G. *Threaded Interpretive Languages.* Peterborough, N.H.: Byte Books, 1981.

DDJ

Vote for your favorite feature/article.
Circle Reader Service **No. 2.**

"How to protect your software by letting people copy it."

By Dick Erett, President of Software Security

Inventor and entrepreneur, Dick Erett, explains his company's view on the protection of intellectual property.

"A crucial point that even sophisticated software development companies and the trade press seem to be missing or ignoring is this:

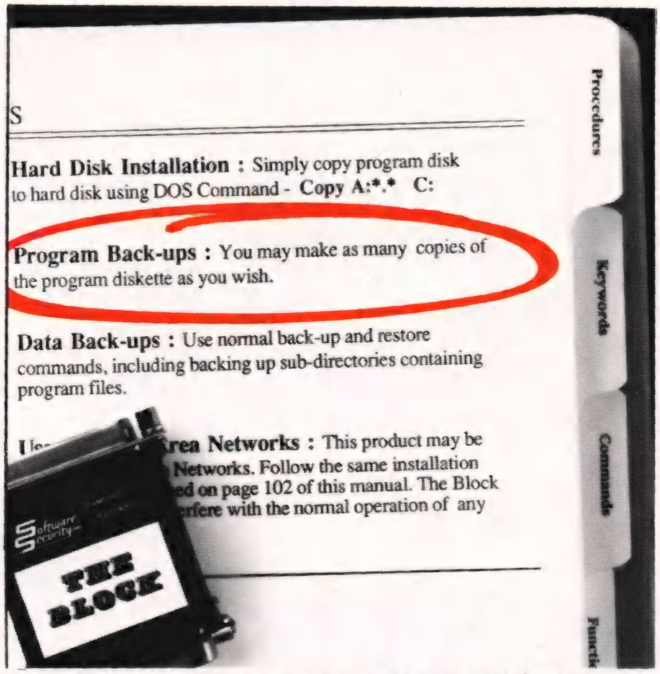
Software protection must be understood to be a distinctively different concept from that commonly referred to as copy protection.

Fundamentally, software protection involves devising a method that prevents unauthorized use of a program, without restricting a legitimate user from making any number of additional copies or preventing program operation via hard disk or LANs.

Logic dictates that magnetic media can no more protect itself from misuse than a padlock can lock itself.

Software protection must reside outside the actual storage media. The technique can then be made as tamper proof as deemed necessary. If one is clever enough, patent law can be brought to bear on the method.

Software protection is at a crossroads and the choices are clear. You can give product away to a segment of the market, or take a stand against the theft of your intellectual property.



Soon all software installation procedures will be as straightforward as this. The only difference will be whether you include the option to steal your product or not.

"...giving your software away is fine..."

We strongly believe that giving your software away is fine, if you make the decision to do so. However, if the public's sense of ethics is determining company policy, then you are no longer in control.

We have patented a device that protects your software while allowing unlimited archival copies and uninhibited use of hard disks and LANs. The name of this product is The BLOCK™.

The BLOCK is the only patented method we know of to protect your investment. It answers all the complaints of reasonable people concerning software protection.

In reality, the only people who could object are those who would like the option of stealing your company's product.

"...eliminating the rationale for copy-busting..."

Since The BLOCK allows a user to make unlimited archival copies the rationale for copy-busting programs is eliminated.

The BLOCK is fully protected by federal patent law rather than the less effective copyright statutes. The law clearly prohibits the production of work-alike devices to replace The BLOCK.

The BLOCK attaches to any communications port of virtually any microcomputer. It comes with a unique customer product number programmed into the circuit.

The BLOCK is transparent to any device attached to the port. Once it is in place users are essentially unaware of its presence. The BLOCK may be daisy-chained to provide security for more than one software package.

Each software developer devises their own procedure for accessing The BLOCK to confirm a legitimate user. If it is not present, then the program can take appropriate action.

"...possibilities... limited only by your imagination..."

The elegance of The BLOCK lies in its simplicity. Once you understand the principle of The BLOCK, hundreds of possibilities will manifest themselves, limited only by your imagination.

Your efforts, investments and intellectual property belong to you, and you have an obligation to protect them. Let us help you safeguard what's rightfully yours. Call today for our brochure, or a demo unit."

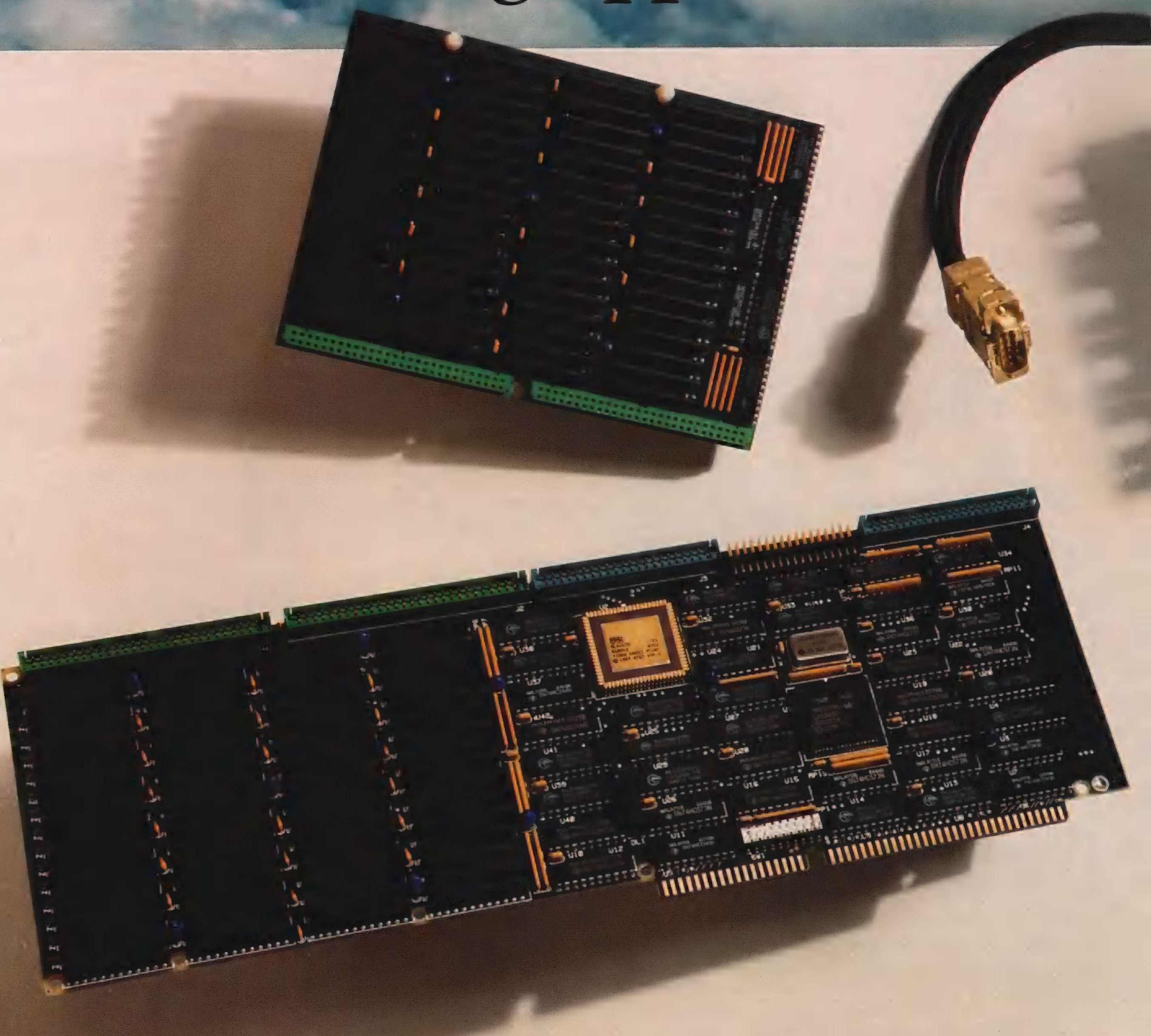
Software Security inc.

870 High Ridge Road Stamford, Connecticut 06905

203 329 8870

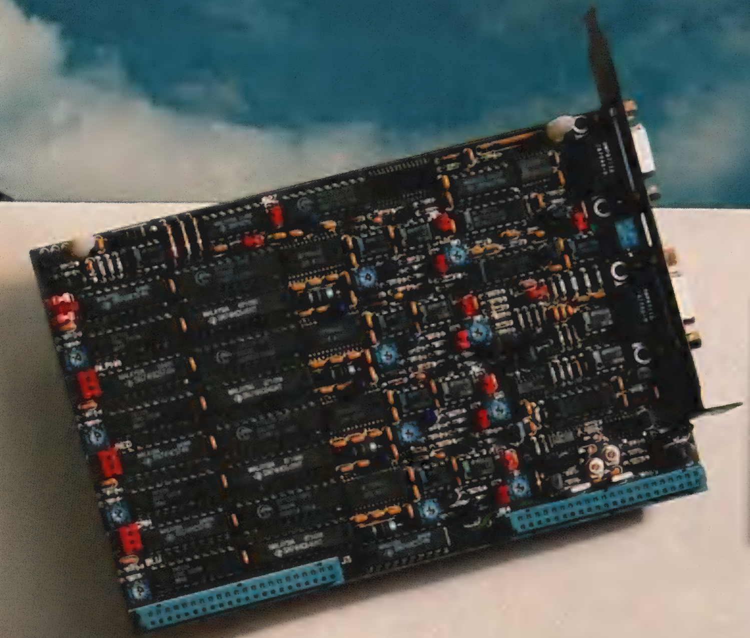
CIRCLE 170 ON READER SERVICE CARD

Now Taking Applications.

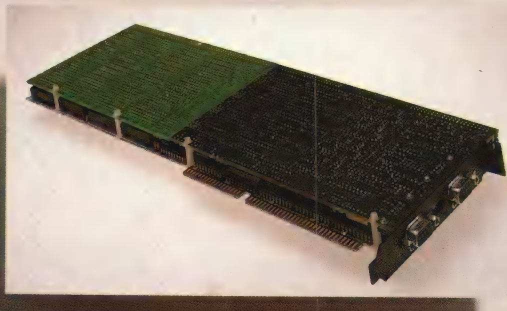


Take a look at the specs on VISTA™, a good look. Notice the processing, programming, and video capabilities? Now think real hard about what *you* could do with the power of VISTA and a microcomputer. Incorporate it with your system to create a digital pre-press proofing station for publishing. Design a graphics workstation which outputs both colorful hi-resolution slides and broadcast-quality animated images. Construct a CAD system which merges computer generated images with real-life backdrops for architecture, packaging or other industries. And, after you've brainstormed your way to new horizons of videographics possibilities, get your own VISTA and start working.

Introducing VISTA™ Videographics.



VISTA is a powerful single-slot videographics adapter which captures and displays video signals in real time.



Let's Get Specific.

We knew you couldn't resist seeing the facts, and frankly, our engineers wouldn't have it any other way. Here is an overview of VISTA's key features.

FEATURES:

- 4Mbytes of Video RAM on-board
- Texas Instruments' TMS 34010 GSP
- Flexible, programmable resolutions
- NTSC and PAL compatible
- Four 8-bit channels for real-time capture
- Fully integrated genlock
- Processor memory expandable in 2Mbyte increments to 12Mbytes
- Four 2K x 8-bit CMOS static RAM LUTs
- Display can be color-mapped, RGB, or a versatile combination of both
- Interlaced and non-interlaced display
- Binary and fractional programmable zoom capability, creates horizontal and vertical magnify or minify
- Smooth horizontal and vertical programmable panning, includes wrap-around and split screen
- Suggested Retail Price: $5995.

ADDRESSABLE RESOLUTIONS:

32 bits/pixel	16 bits/pixel	8 bits/pixel
1024x1024	2048x1024	4096x1024
512x2048	1024x2048	2048x2048
256x4096	512x4096	1024x4096

CAPTURE RESOLUTIONS:*

NTSC	PAL
(RS-170A)	(CCIR-624)
756x486	738x576
604x486	590x576
504x486	492x576
432x486	422x576

Resolutions are programmable; these are nominal ones for interlaced NTSC and PAL compatible.

DISPLAY RESOLUTIONS:*

NTSC	PAL	Interlaced	Non-Interlaced
(RS-170A)	(CCIR-624)		
1512x486	1476x576	1024x768	768x576
1008x486	984x576	(60 Hz)	(50 Hz)
756x486	738x576		
604x486	590x576	768x768	756x486
504x486	492x576	(80 Hz)	(60 Hz)

Resolutions are programmable; these are nominal ones.

COMPUTER REQUIREMENTS:

Host Type:	IBM PC AT and 100% Compatibles, Compaq 386, Apollo DN 3000-single-slot board
Data Bus:	16-bit or 8-bit (self-configuring)
Bus Clock:	6MHz to 12MHz
Power Consumption:	15 Watts

It's So Flexible, We've Added Support.

With its Texas Instruments TMS 34010 graphics processor, large quantity of video memory, and proprietary video cross-point, VISTA can be programmed for an array of powerful market-specific videographic applications. To help you maximize VISTA's potential, Truevision offers a range of C-language programming tools for developers. And when your system is market-ready, we'll support your marketing efforts with our *TRUEVISION SOFTWARE CATALOG, TRUEVISION NEWS,* and *THE PULSE.*

We're For Higher

Resolution…Power…Flexibility…Quality. Join the many key manufacturers and developers already working with the state of the videographics art, VISTA. Call us at **800/858-TRUE** for more information on the VISTA Developer's Program. We're ready to take your application today.

AT&T
Electronic Photography and Imaging Center
7351 Shadeland Station, Suite 100
Indianapolis, IN 46256
800/858-TRUE

AT&T | TRUEVISION®

CIRCLE 294 ON READER SERVICE CARD

IBM is a registered trademark of International Business Machines Corp. **International Inquiries:** contact *Techexport* at 617/890-6507 (USA), or London at 44-1-991-0121. In Italy, contact *SIRIO Informatica* at 39-2-301-0051. Suggested retail price is US domestic price.

A Forth Standard Prelude

by Martin Tracy

Can the Forth-83 Standard dialect be used to write substantial programs? The answer is a qualified "yes." The qualification is that the program must be preceded by a software "prelude." The purpose of this prelude is to provide the program with those facilities that cannot be found within the Standard. The good news is that the prelude can be surprisingly short.

Two years ago, I began editing programs for the Forth Model Library, which is sold by the Forth Interest Group (FIG). The programs had been submitted by different authors and were written in various implementations of the Forth-83 Standard dialect. Each program consisted of 70 to 120 screens of source code. My task was to convert these programs to forms that would run equally well under several available Forth-83 implementations.

I found that a program written in one Forth-83 implementation would not run correctly under another. The reason for this was simple. The authors needed software tools that could not be provided by the Standard but that were instead provided by the implementation. The solution to the problem was also simple. I wrote a software prelude, one for each implementation, that provided the missing functions. The user of a particular Forth-83 package then simply loaded these screens before loading the program.

When I say "could not be provided by the Standard," I do not mean words that do not happen to appear in the Standard but that can be written using Standard words. Take, for example, the words *NIP* and *TUCK*, which can be found in the Laxen/Perry F83 implementation:

```
: NIP ( n n2 — n2)       SWAP DROP ;
: TUCK ( n n2 — n2 n n2) SWAP
                               OVER ;
```

Forth recurses faster and more naturally than any other popular language.

As you can see, these words can be defined using the Forth-83 Standard words *SWAP DROP* and *OVER*. Thus, although *NIP* and *TUCK* are not mentioned in the Standard, they can be written with Standard words and used in Standard programs. It is up to the author (or the hapless editor) to make sure these words are defined before use.

Recursion

Consider, on the other hand, recursive definitions. The Forth-83 Standard does not mention recursion in its Required Word Set, although it defines *RECURSE* in the Controlled Reference Word Set:

```
RECURSE  --       C,I,83
         -- (compiling)
```

"Compile the compilation address of the definition being compiled to cause the definition to later be executed recursively."

Controlled reference words are word definitions that "although not required, cannot be present with a non-standard definition in the vocabulary FORTH of a Standard System."

You would know what *RECURSE* meant if you saw it in a Standard program. This does not mean, however, that programs using recursion must do so with *RECURSE*. Some implementations prefer the word *MYSELF*. Others, like Laxen/Perry F83, provide a different mechanism entirely: words whose definitions begin with the command *RECURSIVE* can refer to themselves.

In other words, the Standard says "a word by this name has this function and syntax" rather than "this function is performed by a word of this name and syntax." The confusion of form and function is apparent in other places as well. Consider the definition of *>NAME* in the "Experimental Proposal: Definition Field Address Conversion Operators":

>NAME addr1 – addr2 "to-name"
"addr2 is the name field address corresponding to the compilation address addr1."

The form or syntax of this word is the same as its function: to take you from the compilation address to the name field address. Because you don't know the structure of the name field, however, you can't do anything with it. The underlying function you really want is to display the name of a word, given its compilation address.

Is recursion important? You bet. Forth recurses faster and more naturally than any other popular high-level programming language, including C and LISP. Why? First of all, because

Martin is DDJ's Forth columnist. He can be reached at Forth Inc., 111 N. Sepulveda, Manhattan Beach, CA 90266.

Forth words keep the majority of their arguments on the stack, which is a naturally recursive structure. Second, because Forth generally does not use local variables and so has no stack frame or other lexical structure to build each time it recurses.

A popular belief, even among Forth programmers, is that recursion should be avoided because the return stack is small. The Standard guarantees a return stack of only 48 bytes. But this only means that "tail-recursive" problems, such as those that visit each item on a list, should be unravelled into iterative structures, such as a *DO* loop. (Modern LISP compilers do this automatically.) There is an equally rich set of "head-recursive" problems, however, that recurse only to the depth of the problem and not to its breadth. These are problems such as shape-filling algorithms, tree traversal, and natural language parsing, which seldom recurse any deeper than eight or nine levels.

Fortunately, *RECURSE* can be defined easily even in implementations that do not include it. For example:

```
: RECURSE   [COMPILE] MYSELF ;
                        IMMEDIATE
```

This definition is likely to be a "one-liner" even in implementations that support some other recursion mechanism entirely. Why? Because the concept of recursion is simple and natural to the Forth language. Simplicity and harmony are the guidelines for selecting words for the Standard prelude.

Compiler Words

The Forth-83 Standard provides a set of words to support the compiler, such as *[COMPILE]* and *IMMEDIATE.* These words can also be used to extend the compiler. They are, in fact, the building blocks for new compiler words. Compiler words are used in Forth to build flow-of-control structures and to "hide and provide" in-line data in colon definitions. The extensible compiler is one of the true strengths of the Forth language.

Compiler words generally have two functions: they must compile both a run-time operator and the in-line data upon which it operates. The run-time operator also has two functions: it must operate meaningfully on the in-line data, and it must adjust the Forth instruction pointer (which I will call *I*) to skip over this data.

Unfortunately, the Standard does not provide for the creation of new run-time words. A run-time word has no Standard way of finding or skipping over the following in-line data. Consider how the word *LITERAL* might be defined:

```
: lit ( — n) R> DUP @ SWAP 2+ >R ;
: LITERAL   COMPILE lit , ;
                        IMMEDIATE
```

LITERAL compiles the run-time word *lit*, followed by the number on the stack. Because *lit* is a colon definition, you might expect *R>* to move *lit*'s return address to the data stack. Furthermore, the return address of *lit* should be the address of the in-line data that follows, right? Actually, this definition of *lit* will work correctly on a great many Forth implementations. Not all Forths increment the instruction pointer *I* to point to the next address, though. Some increment it only on demand, reasoning that the increment is wasted when it precedes a branch. Others compromise and only increment it to point to the next byte. In these cases, *R>* points near to but not directly at the in-line data and must be adjusted. The adjustment can be hidden in the way shown in Example 1, below. Experience has shown that this solves the problem in almost all cases. *I>* (*I*-from) and *>I* (to-*I*) have even been written for a Forth with a 16-bit data stack width and a 32-bit return stack width and have worked correctly.

Alignment

Now consider an often requested function—the in-line string compiler. Usually called simply *"* (double quote), it might be implemented in this way:

```
: (") ( — addr n)   I> @ COUNT 2DUP
                              + >I ;
: "    COMPILE (") 34 WORD
              DUP C@ ( n ) 1+ >R
       COUNT HERE SWAP CMOVE R>
              ALLOT ; IMMEDIATE
: EXAMPLE
   " EXAMPLE prints a string." TYPE ;
```

This definition of *"* works, in principle, on strings with an odd number of characters. Otherwise, on a byte-addressed machine with even address or "cell" alignment, the definition fails. On some Motorola 68000 Forth implementations, the failure is fatal.

You can easily define a pair of words to hide address alignment, provided you are able to make one simplifying assumption: Forths that

```
: I> COMPILE R> ; IMMEDIATE ( no offset)
: >I COMPILE >R ; IMMEDIATE ( no offset)

: I> R> R> 1+ SWAP >R ; IMMEDIATE ( one-byte offset)
: >I R> SWAP 1- >R >R ; IMMEDIATE ( one-byte offset)

: I> R> R> 2+ SWAP >R ; IMMEDIATE ( two-byte offset)
: >I R> SWAP 2- >R >R ; IMMEDIATE ( two-byte offset)

: lit ( — n) I> DUP @ SWAP 2+ >I ;

: LITERAL COMPILE lit , ; IMMEDIATE
```

Example 1: *A way to hide the adjustment of* R>

```
: ALIGN ; IMMEDIATE ( no alignment)
: REALIGN ;    ( no alignment)

: ALIGN HERE 1 AND ALLOT ; IMMEDIATE ( cell-alignment)
: REALIGN ( a — a') DUP 1 AND + ; ( cell-alignment)

: (") ( — addr n) I> @ COUNT 2DUP + REALIGN >I ;

: " COMPILE (") 34 WORD DUP C@ ( n ) 1+ >R
  COUNT HERE SWAP CMOVE R> ALLOT ALIGN ; IMMEDIATE
```

Example 2: *A definition of " when the dictionary is aligned*

THE ADA® WORLD HAS CHANGED.™

Next Generation Ada Technology

Third generation Ada is here today with Meridian AdaVantage, and it's validated by the Department of Defense and their suite of 2,700 tests. Ada's first generation was the proving ground technology of the early eighties. Then came the large validated compilers of the mid-eighties, which were often inefficient and machine specific. Now Meridian offers a compact efficient Ada technology that is highly portable from PC's to minis to mainframes.

COMPACT MERIDIAN Ada IMPLEMENTATION

The chart above depicts the dramatic difference between Meridian's implementation and other comercially available Ada implementations. Meridian's order of magnitude smaller code size results in an extremely fast compiler that will produce highly optimized code.

Meridian's Track Record

Since 1981, we've been building compilers the old-fashioned way, through hard work and experience gained from our development of successful portable compiler products. These products are installed at over 1,500 sites, including almost all major DoD contractors as well as commercial software developers.

VALIDATED Ada

THIS PRODUCT CONFORMS TO ANSI/MIL-STD-1815A AS DETERMINED BY THE AJPO UNDER ITS CURRENT TESTING PROCEDURES

Price/Performance Breakthrough

The Meridian AdaVantage v2.0 validated Ada compiler costs $795 and provides a production quality Ada development tool.

SIEVE BENCHMARK

	Meridian v2.0	Alsys v1.3
Compile and link time	27 sec	59 sec
Execution time	4.6 sec	4.9 sec
Execution size	27,344 bytes	42,129 bytes
Price	$795	$2,995

NOTE: All times measured on an IBM at 5170 (8MHz) and a 4MB RAM card required by the Alsys system. When running without the RAM card, the Meridian compile and link time is 46 seconds.

The Meridian AdaStarter incorporates all of the features of AdaVantage, with certain size limitations. AdaStarter is perfect for anyone who wants to learn how to program in Ada. The complete $99 cost is applicable toward the purchase price of the AdaVantage production compiler.

The compilers all run in a standard PC configuration with 640K of memory, a hard disk, and DOS v2.1 or higher.

To order today, or get more information, call 1-800-221-2522 (outside California) and 1-714-380-9800 (inside California).

"The Meridian compiler is a very well-thought-out compiler. The compiler is fast and execution speed is more than adequate... Overall, we'd rate the Meridian compiler as very solid."
— COMPUTER LANGUAGE, DECEMBER 1986

"The more affordable AdaVantage v1.0 is good for the average programmer because of its price, the extent of its implementation, and its relaxed hardware requirements... [AdaVantage v2.0] should be competitive with Alsys Ada, since it will be a full Ada at less than a third of the price of Alsys."
BYTE, JULY 1987

MERIDIAN
SOFTWARE SYSTEMS, INC.

23141 Verdugo Drive, Suite 105, Laguna Hills, CA 92653
800/221-2522 (outside CA) 714/380-9800 (inside CA)
Telex: 650-268-0547 MCI Fax: 714/380-1683

FORTH PRELUDE

(continued from page 41)

use cell alignment keep the dictionary in an aligned state at all times. Even if the dictionary is only aligned while compiling a definition, you can define *"* with *ALIGN* and *REALIGN* as shown in Example 2, page 41.

You might argue that the word *"* should be in the Standard prelude instead of the *ALIGN* and *REALIGN* pair. The trouble with *"* is that it's not simple enough. It leaves you with no way to compile a sequence of nonprintable control characters or to align a *CREATE . . . DOES>* structure. Furthermore, there is no general agreement among implementors as to whether *"* should return the one-argument address of a counted string or the two-argument address of the first character and its length—a form suitable for *TYPE*. By redefining *"* in the Standard prelude, you can guarantee its syntax.

Interpreting a String

The Forth-83 Standard describes the terminal input buffer in such a way that you might think that if you *CMOVE* a string into *TIB*, store its length into *#TIB*, and set *BLK* and *>IN* to 0, then you will force the Forth text interpreter to interpret the string. Why would you want to do that? Well, it is handy to be able to compile a word such as *FIND* or *FORGET* in a colon definition along with its argument.

Unfortunately, these words are defined to read their arguments from the input stream. How nice if you could compile the input stream as well:

```
: ME ( Initialize system, then. . .)
                    " FORGET ME" EVAL ;
```

I am assuming that *EVAL* does the work of moving the string into *TIB* and so on. In the same manner, you could use *FIND* to see if a particular word is present in the dictionary:

```
: WELL?   " FIND ME" EVAL IF . . .
```

You could also create words from within other words and reference words before they are defined. The fundamental right to treat (string) data as an executable program is guaranteed by the Von Neumann architecture.

I have borrowed the word *EVAL* from the LISP function by the same name. The problem with *EVAL* is that it's not simple enough. The Standard already lets you set up *TIB #TIB >IN* and *BLK*. But it provides you with no way to invoke the text interpreter to interpret it. The function you are missing is *INTERPRET*. Like *RECURSE*, it is found in the Controlled Reference Word Set:

INTERPRET -- M,83
"Begin text interpretation at the character indexed by the contents of >IN relative to the block number contained in BLK, continuing until the input stream is exhausted. If BLK contains zero, interpret characters from the text input buffer."

Given *INTERPRET*, you can now define a simple *EVAL*:

```
: EVAL ( a n)   DUP >R TIB SWAP
                   CMOVE R@ #TIB !
  0 >IN ! 0 BLK ! INTERPRET R> >IN
                                 ! ;
```

The sequence *R> >IN !* marks the input stream as exhausted. I have chosen the two-argument string form. With *INTERPRET*, you can implement either string form.

Screen Display

The six words *RECURSE*, *INTERPRET*, *I>*, *>I*, *ALIGN*, and *REALIGN* supply the most often requested non-Standard functions. The most often requested extension, however, is for video screen control. Virtually all available Forth-83 implementations allow the programmer to control the appearance and cursor position of the video display. A smart presentation can mean more to a program than a string stack or floating-point math.

Although video displays vary widely, modern displays have been standardized to a gratifying extent. You can, in fact, add a fairly good screen display extension to the Standard prelude if you follow a few simplifying assumptions and rules:

1. Assume the screen is at least 80 characters wide and 24 lines high.
2. Define the word *PAGE* to clear the entire screen and home the cursor to

FULL AT&T C++ for half the price of our competitors!

Guidelines announces its port of **version 1.1** of AT&T's C++ translator. As an object-oriented language, C++ includes: classes, inheritance, member functions, constructors and destructors, data hiding, and data abstraction. 'Object-oriented' means that C++ code is more readable, more reliable and more reusable. And that means faster development, easier maintenance, and the ability to handle more complex projects. C++ is **Bell Labs' answer to Ada and Modula 2**. C++ will more than pay for itself in saved development time on your next project.

C++

from GUIDELINES for the IBM PC: $195

Requires IBM PC/XT/AT or compatible with 640K and a hard disk.
Note: C++ is a *translator*, and requires the use of Microsoft C 3.0 or later.

Here is what you get for $195:
- The full AT&T v1.1 C++ translator.
- Libraries for stream I/O and complex math.
- "The C++ Programming Language", the definitive 327-page tutorial and description by Bjarne Stroustrup, designer of C++.
- Sample programs written in C++.
- Installation guide and documentation.
- 30 day money back guarantee.

To order:
send check or money order to:

GUIDELINES SOFTWARE
P.O. Box 749
Orinda, CA 94563

To order with Visa or MC, phone (415) 254-9393.
(CA residents add 6% tax.)

C++ is ported to the PC by Guidelines under license from AT&T.
Call or write for a free C++ information package.

 CIRCLE 351 ON READER SERVICE CARD

A Different BASIC Might Make All the Difference

We'll skip the four-color gatefold. And the extravagant claims.
Because if you're serious about programming, you just want the straight facts:

	True BASIC 2.01	Microsoft Quick Basic 3.0	Borland Turbo Basic 2.0
GRAPHICS			
Supports Hercules Graphics	YES	NO	NO
Device-Independent Graphics Syntax	YES	NO	NO
User-Defined Coordinates	YES	LIMITED	LIMITED
Matrix Graphics Coordinates	YES	NO	NO
ARRAY HANDLING			
Matrix Algebra	YES	NO	NO
Maximum Numeric Array	UNLIMITED	64K	64K
Max. Number of Array Dimensions	255	63	8
Max. Number of Elements/Dimension	UNLIMITED	32K	32K
Dynamic Redimensioning	YES	NO	NO
Matrix I/O Statements	YES	NO	NO
STRING/FILE HANDLING			
Maximum String Length	64K	32K	32K
Total String Space	UNLIMITED	64K	64K
Maximum Record Size	16MB	32K	32K
Max. Bytes/Binary File Read	64K	NA	32K
PRODUCTIVITY ENHANCERS			
Modules	YES	NO	NO
Separately Compiled Libraries	YES	LIMITED	NO
Workspaces	YES	NO	NO
Immediate Mode	YES	NO	NO
SPECIAL FEATURES			
Stop/Continue Execution	YES	NO	NO
Max. Source File	UNLIMITED	UNLIMITED	64K
Script Files	YES	NO	NO
Keystroke Macros	YES	NO	NO
Max. Characters/Line	64K	255 char.	249 char.
Max. Scalar Data Space	UNLIMITED	64K	64K
Mouse Support	YES	NO	NO
80386 Version	YES	NO	NO
Portability to:	Macintosh, Amiga, Atari	Translation required	No other machines

NOW ONLY $99.95

Three very structured, very powerful programming packages. All with fancy editors and fast compilers. Two of them are the same in other respects. And one of them, True BASIC, is quite a bit different. With syntax and features that will make you more productive.

That's why reviewers for magazines like BYTE, PC Tech Journal and Computerworld keep giving True BASIC their top marks. And why OEMs pick True BASIC after they've evaluated all the others. See why True BASIC can make the difference for you.

Call **1-800-TRBASIC** today.

True BASIC

True BASIC, Quick Basic and Turbo Basic are trademarks of True BASIC, Inc, Microsoft and Borland, respectively. Macintosh, Amiga and Atari are trademarks of Apple Computer, Inc., Commodore-Amiga, Inc. and Atari Corporation. Copyright 1987 True BASIC. Specifications are accurate as of August 1987.

39 SOUTH MAIN STREET
HANOVER, N.H. 03755
(603) 643-3882

CIRCLE 344 ON READER SERVICE CARD

FORTH PRELUDE
(continued from page 43)

the upper left-hand corner.

3. Define the word *TAB (x y)* to position the cursor at the given *x* (character) and *y* (line) coordinates. Coordinate pair (0,0) is in the upper left-hand corner.
4. Define the word *MARK (a n)* to print the given two-argument character string in highlight or inverse video.
5. Never write into or over column 79. Never issue a carriage return from row 23. In other words, you do not face the issues of wrapping the line or scrolling the display.

It turns out that these five restrictions are sufficient to generate some really nice displays.

Cell Addressing

The Forth-83 Standard describes only implementations on byte-addressed CPUs with a 64K address space. Future Forth standards are likely to consider much larger address spaces. There are already several 32-bit Forth implementations.

A Standard Forth implementation assumes that there are 2 bytes per cell, and Standard programs are filled with 2+s and 2*s accordingly. On a byte-addressed 32-bit Motorola 68000 implementation, however, there are 4 bytes per cell. On a cell-addressed Novix 4016 or Texas Instruments TMS32020, there may be only 1 byte per cell. The number of bytes per cell is used mostly to specify how much dictionary memory to allocate or how to skip to the next cell of a data structure.

It can be hidden from an application with *CELL*, *CELLS*, and *CELL+* as shown in Example 3, page 45.

Byte order within a cell is normally not a problem. Bytes that are written by byte operators should be read by byte operators. Be careful when you define byte operators that are based on cell operators to make them independent of the byte order.

An Experimental Proposal

The Forth community has long been searching for a solution to a classic programming problem: the string search. When you search for characters in a string, you generally use a

DO loop. You leave the loop in one of two circumstances:

1. The search is successful. You leave the loop immediately.
2. The search is unsuccessful. You leave the loop because it is exhausted.

The problem is that once you have left the loop, how do you know if the search was successful?

One solution is to maintain a flag on the stack:

```
: SEARCH . . . 0 ( flag ) ROT ROT
  DO DROP . . . ( compare strings ) =
    IF . . . -1 LEAVE THEN 0 ( flag )
  LOOP ;
```

If the search is successful, the flag will be true.

Leo Brodie, Wil Baden, and others point out that a much better approach is to leave the loop and the word that contains it when the search is successful:

```
: SEARCH . . .
  DO . . . ( compare strings ) =
    IF . . . -1 LEAP ( leave word entirely) THEN
  LOOP 0 ;
```

LEAP is Leo Brodie's suggestion, but it has the usual problem: it's not simple enough.

Wil Baden's solution looks like this:

```
: SEARCH . . .
  DO . . . ( compare strings ) =
    IF . . . -1 UNDO EXIT ( leave word entirely) THEN
  LOOP 0 ;
```

The command *UNDO* "undoes" the loop by discarding the index, limit, and any other loop items on the return stack before leaving the word with *EXIT*. *UNDO* has the additional charm that it can leave a word from a nested loop, as in *UNDO UNDO EXIT*.

UNDO could be defined as a colon definition in this way:

```
: UNDO I> R> R> 2DROP >I ; (discard 2 items)
: UNDO I> R> R> 2DROP R> DROP >I ; ( discard 3 items)
```

Actually, *UNDO* is more easily defined as a *CODE* definition and in some implementations is only one instruction.

In Summary

The Forth Standard prelude described in this article (see Listing One, page 90) has proven to be an effective way to write substantial programs that run unchanged on several different implementations of the Forth-83 Standard. The prelude can hide differences in byte addressing and cell alignment from the application programmer. As more experience is gained, it may be possible to extend the prelude to hide ROM-ability and other implementation dependencies.

```
2 CONSTANT CELL ( byte-addressed 16-bit cells)
: CELLS 2* ; ( size of cell area in bytes)
: CELL+ 2+ ; ( skip to the next cell address)

4 CONSTANT CELL ( byte-addressed 32-bit cells)
: CELLS 4 * ; ( size of cell area in bytes)
: CELL+ 4 + ; ( skip to the next cell address)

1 CONSTANT CELL ( cell-addressed one-byte-per-cell)
: CELLS ; ( size of cell area in bytes)
: CELL+ 1+ ; ( skip to the next cell address)

CELL ALLOT ( allocate a cell)
10 CELLS ALLOT ( allocate 10 cells)

( addr ) DO . . . CELL+ ( skip to next cell) LOOP
```

Example 3: *A way to hide the number of bytes per cell*

Availability

All the source code for articles in this issue is available on a single disk. To order, send $14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call (415)366-3600.

Bibliography

Baden, Wil. "Escaping Forth." *1986 FORML Proceedings*. Available from the Forth Interest Group ([408] 277-0668).

Brodie, Leo. *Thinking Forth*. Englewood Cliffs, N.J.: Prentice-Hall, 1984.

DDJ

(Listing begins on page 90.)

Vote for your favorite feature/article.
Circle Reader Service **No. 3.**

NEW!

Documentation is a PAIN !

. . . without DocuMotion™

We've found that well indexed and easily accessed documentation is a powerful tool and asset. Now you can simply pop up **DocuMotion** to access, display and print **your** documentation. **DocuMotion** builds indexed document libraries from documentation contained in your source code or any text file.

DocuMotion for programmers:

- Your documentation is available ANYWHERE, ANY TIME.
- Access, display and print your documentation by name or by user-defined categorization trees.
- 19 function Microsoft Windows-style menu bar.
- Powerful print & copy functions.

DocuMotion for project mgrs:

- Programmers produce more and better documentation.
- Reduced function redundancy.
- Greater programmer productivity.

DocuMotion for the PC:

- Runs memory resident or non-resident on any IBM PC/XT/AT or compatible.
- Compatible with all popular memory resident programs.
- Requires only 93K RAM.
- LAN compatible.

DocuMotion is for YOU:

Call NOW 1-612-884-5860

At a special introductory price of ONLY **$159.95** with ANSI 'C' document library.

Demo disk for $24.95 that puts the ANSI 'C' library functions on-line.

NWP - Intelligent Solutions, Inc. P.O. Box 20478 Bloomington, MN. 55420-0478

CIRCLE 400 ON READER SERVICE CARD

Pattern Matching Using Finite State Machines

by Charles F. Bowman

If you're a programmer, the chances are you've had to deal with the problem of pattern matching. It can be a simple problem solved with a trivial string comparison utility, or it can be so complex as to require the use of a lexical analyzer.

Most people deal with pattern matching every day: as part of a search-and-replace operation in a text editor, as a means of data retrieval in a database application, as unique identifiers in program source code, and so on. Because the task is so common, it's worthwhile to examine ways to perform it as efficiently as possible. This article describes how state machines can be useful in solving this type of problem.

To demonstrate the use and effectiveness of state machines, this article provides the source code for a program called findcmd. The program does just what the name implies—it finds commands. When invoked, the program searches each component of the user's *path* variable for all programs (files) that match the supplied pattern arguments. The pattern string can contain wildcard characters just like those the DOS command shell accepts. The program uses an extension of the Knuth-Morris-Pratt (KMP) algorithm, which implements pattern matching using a finite state machine—more on this later.

State Machines

A complete and formal discussion of Turing machines, automata theory, and so on is beyond the scope of this article. For a more detailed discussion of the topic, see the bibliography.

The program searches each component of the path variable for all files that match the pattern.

In general, a state machine has the following attributes:

- A finite set of states, including an initial, or start, state and a stopping, or end, state.
- A finite set of state transitions (a collection of moves the machine can make). Generally, the transitions are represented as a two-dimensional array, indexed on one axis by the current state number and on the other axis by the current input token.
- A set of input symbols (alphabetical) on which the state transitions will occur (for my purposes, I define this as the file names that will be compared with the pattern arguments).
- A read head that points to the next available input character.

The basic operation of a state machine can be described as follows. Upon invocation, the machine is initialized to the begin state. It then iteratively examines the current input symbol and, using the set of state transitions, progresses or moves to the next state, advancing the read head as required. It continues this process until it encounters either an error condition or an accepting state. The actual transition decisions are accomplished by what is effectively a table lookup. Both the current state and current input token are used to index into the transition table to determine the resulting action of the machine. The range of actions includes stop (error), accept, and move (to new state).

The stop action occurs when the current input token is invalid with respect to the current state. This situation raises an error condition that usually results in the machine halting (in a compiler, for example, this would typically be the point at which you would receive a message such as "syntax error line 16"). The accept action occurs only when the machine is in a valid halt state and has exhausted the input stream (the read head points to the end of a file). (In a compiler, this would mean your source code was syntactically correct.) The intermediate transition states comprise the remainder of the table; they move the machine from state to state based on the input stream.

There are two important points to note here. One is that the machine is not required to absorb an input symbol (advance the read head) with each state transition. It is perfectly acceptable for the machine to accomplish multiple state transitions with the same input symbol remaining current. The other point is that, by the very virtue of the state transitions, the machine always "knows" what it has seen previously. In other words, you could say, "If the machine is in state X, then the last N characters have to be the following . . . ." This is an extremely important characteristic of a state machine because it affords it the luxury of discarding input symbols once it has used them. (If need be, the

Charles F. Bowman, 24 Jacques Ave., Staten Island, NY 10306. Charles is a consultant and is currently writing a textbook on data structures. He holds an M.S. degree from New York University.

machine can reconstruct the input stream for the last N characters just from knowing the current state.)

Uses of State Machines

State machines have many uses. The most common, as mentioned earlier, is in compiler writing, where they are typically used in the lexical analysis and the parsing phases of compilation. There are also programs (most notably YACC) that produce a state machine from a formal definition of a language. Database management systems (query languages) also rely heavily upon state machines.

I have used state machines on numerous occasions, most recently when I was asked to write a program that would extract information selectively from a continuous, on-line data stream. The input was being produced by a PBX that generated call-usage and call-summary reports. The application required that only selected data, from a subset of the reports, be extracted and stored for further processing. The obvious difficulty was remembering, at any given point, which of the many reports were being read and what data to extract. I was able to write such a program quickly and efficiently by modeling the events within the framework of a state machine.

KMP Algorithm

The Knuth-Morris-Pratt (KMP) algorithm accomplishes pattern matching through the use of a state machine. Using this technique, you can efficiently construct a finite automaton for a given pattern string. Moreover, you can then use the machine to test quickly for an occurrence of the pattern in subject strings. The algorithm is really divided into two sections: the first produces the state machine (transition table) derived from its pattern arguments; the second compares the compiled pattern with subject strings.

The transition table is constructed in a straightforward manner. It has an initial or start state, followed by one or more transition states that are derived directly from the pattern string. A one-to-one correspondence exists between the pattern characters and the generated machine states. These are then followed by an accept state. Refer to the function *inits( )* in Listing Three, page 106, for an example.

The second part of the algorithm uses the transition table to make comparisons with input strings. It begins processing in the start state and then iteratively compares the current state information with the corresponding subject string character. If the comparison shows them to be equal, the machine moves to the next state; if they are not equal, the strings are not identical and the machine halts. If the machine reaches the accept state at the same time as it exhausts the subject string, it halts and accepts the input (the pattern and subject match).

Let's take a close look at the operation of the algorithm. For this example, assume a pattern string $P = p_1 p_2 p_3 \ldots p_n$ and an input string $I = i_1 i_2 i_3 \ldots i_n$. The machine begins in state 0 with its read head pointing at i_1.

If the first input token, i_1, is not equal to p_1, then the machine remains in state 0. If $i_1 = p_1$, then the machine advances to state 1. In both cases, the read head is advanced and the current input token is discarded. The machine continues in this fashion as long as the current input symbol matches the pattern character of the current state.

To generalize, suppose that, after having read the input symbols $i_1 i_2 i_3 \ldots i_k$, the machine is in state j. That means that the last j tokens of the input stream are equal to $p_1 p_2 p_3 \ldots p_j$, the first j tokens of the pattern string. If $i_{k+1} = p_{j+1}$, then the machine enters state j+1 and advances the read head. If $i_{k+1} <> p_{j+1}$, then the machine must recover—that is, it must begin to look for an occurrence of the pattern string at the next logical input position. It cannot however, just blindly enter state 0 and resume processing with the next (current) input token; it could miss an occurrence of the pattern beginning at locations $i_{(k-j)+1}$ or $i_{(k-j)+2}$. And, although possible (as stated previously), it is extremely inefficient to have the machine reconstruct and reprocess portions of the input stream. What the machine must do, therefore, is "shift" the pattern forward so that it lines up with some portion of the input stream already processed.

Figure 1, below, shows an example of how this works. If the next subject character, X, is not an A, then the state machine would move the pattern forward as in Figure 2, below. Consequently, it saves the expense of having to compare the pattern beginning at the first B in the subject string. Also, because the states effectively "remember" the last n symbols, the machine is not forced to backtrack over the input stream. (As mentioned earlier, this feature lets the machine discard input tokens once it has read them.)

To accomplish this algorithmically, the machine employs a failure function, f(j), which is defined as returning the largest s (smaller than j) such that $p_1 p_2 p_3 \ldots p_s$ is a suffix of $p_{j-s+1} p_{j-s+2} p_{j-s+3} \ldots p_j$. That is, $p_1 p_2 p_3 \ldots p_s = p_{j-s+1} p_{j-s+2} p_{j-s+3} \ldots p_j$.

Before I demonstrate how to compute the failure function, I'll explain how it will be used. Given the pattern string P=aabbaab, the values of the failure function will be as shown in Figure 3, below.

Suppose that the machine is again in state j, having read $i_1 i_2 i_3 \ldots i_k$. Further, suppose that $i_{k+1} <> p_{j+1}$. The machine will apply the failure function in the following manner:

```
Pattern:       |ABCABC|   A . . .
Subject:    . . |ABCABC|   X
```

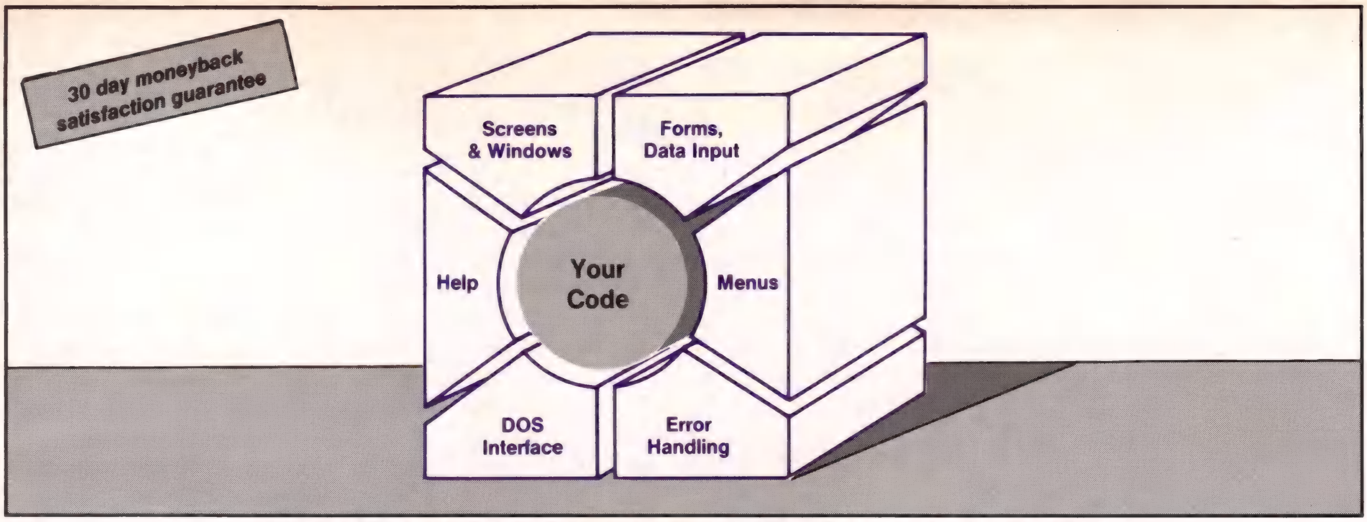
Figure 1: *Before the comparison*

```
Pattern:             |  ABC  |  ABCA . . .
Subject:   . . . ABC|  ABC  |  X . . .
```

Figure 2: *After the comparison*

```
STATE      | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
--------   | --- | --- | --- | --- | --- | --- | --- |
F(STATE)   | 0 | 1 | 0 | 0 | 1 | 2 | 3 |
```

Figure 3: *Values of the failure function for the pattern string P=aabbaab*



C-Worthy Interface Library helps you smoothly pull together all aspects of an excellent Human Interface.

C Programmers: Wrap an Exciting, Bullet-Proof Interface Around Your Code Quickly.

Introducing... C-Worthy® Interface Library

The only human interface package you need. That's what our customers are telling us. One early adopter, Novell, Inc. uses it exclusively in the development of their NetWare® Utilities, which reach over 500,000 users. You see, C-Worthy Interface Library is the only library available to handle every aspect of your program's human interface, all in one package. Now your programs will have a consistent look and feel. You no longer have to integrate pieces of libraries from different manufacturers.

As important as you know users are, you often don't have the time to heavily invest in writing routine code. And that's OK, because with over 400 tight, ready-to-use functions, C-Worthy Interface Library takes care of the tedium and lets you spend your time doing what you enjoy. Concentrate on the heart of your application — features that make it unique, special. Let C-Worthy Interface Library do your:

- Menus
- Error Handling
- DOS Interface
- Context Sensitive Help
- Screens, Windows
- Forms, Data Input (optional)

You control color, size, border, location, etc. And if there's anything you want to change, you can. Source is available to provide you with the flexibility you need. And you can distribute your applications freely, with no royalties.

C-Worthy Interface Library requires hard disk media with 256K RAM. MSDOS 2.0 + and IBM PC, or compatible, TI Professional, NEC APC III, or VICTOR 9000. C-Worthy is a registered trademark of Custom Design Systems, Inc.

Tech Specs

- Compilers: Microsoft 3.0+, Quick, Turbo, Lattice. All models.
- 350+ functions written in C, 75+ in Assembler.
- Menus: Fully support pop-up, Lotus style, MS Windows style (pull-down), pull-up.
- Errors: DOS, program, and user.
- DOS Interface: 62 functions. File handling, dir. and drive management, date & time conversion, wildcards, more.
- Help: System and context sensitive.
- Screens: Screen display, color palettes, save, restore, scroll, more.
- Windows: Exploding, tiled, pop-up, overlapping. Direct video access and virtual. Up to 50 active at any time.
- Keyboard Handling: Regular, function, interrupt, background procedures.
- Editing: String and word wrap text.
- Form Interface Library: 118 functions. Over 15 field types, and user definable field types. 3 levels of data validation: type, multiple field ranges, optional validation procedures. Hide, lock, or secure a field. Optimal field movement.
- Foreign Languages: All text messages in separate files for easy translation.
- Compatible with MS Windows.
- OS/2 special overlay when released.
- Machines: Autodetect for MDA, CGA, EGA, VGA, TI, AT&T, Victor.
- No royalties.

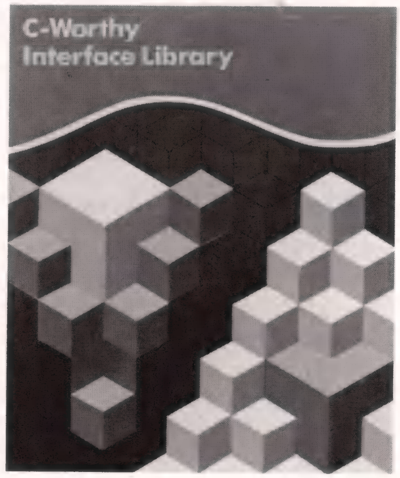
"I heartily recommend this package."
— David A. Schmitt, president, Lattice, Inc.
Over 400 developers in 16 countries already use it.

Thorough Documentation

Indexed alphabetically and by category, the 700+ page Reference Guide includes for each function: an example, description, calling conventions, return values, and related functions. The 250 page User's Guide gets you going with its tutorial and "Getting Started" sections.

CIRCLE 296 ON READER SERVICE CARD

"C-Worthy is a comprehensive C library whose time has come. I heartily recommend it as your next purchase." —Computer Language, 8/87



C-Worthy Interface Library:

Object only	$ 195
Form Interface Library add-on	$ 100
Object with Forms	$ 295
Object with Forms & Library Source	$ 495

Please specify compiler and version when ordering.

To Order Call
(800) 821- 2492
in MA (617) 337-6963

Solution Systems™
541-D Main Street, Suite 410
South Weymouth, MA 02190

(continued from page 47)

Step 1: If u = f(j) and $i_{k+1} = p_{u+1}$, the machine enters state u+1 and advances the read head.
Step 2: If f(j) <> 0, then j = f(j) and repeat step 1.
Step 3: If f(j) = 0 and i_{k+1} <> p_1, then the state is reset to 0 and the read head is advanced.

For example, given a pattern P=aabbcc and an input I=aabbaabbcc, the machine would undergo the sequence of transitions shown in Figure 4, right. Notice that in step 5 the pattern was shifted to the third a of the input stream, where the search was resumed.

The failure function is implemented as a table that is created in a manner analogous to its use. It begins with f(1) = 0, by definition. The next steps are easiest to explain by way of an example.

Suppose you have computed f(1), f(2), f(3), . . ., f(j) and that f(j) = i. To compute f(j+1), you compare p_{j+1} with p_{i+1}. If they are equal, then f(j+1) = f(j) + 1. This is because, $p_1 p_2 p_3 \ldots p_i p_{i+1} = p_{j-i+1} p_{j-i+2} p_{j-i+3} \ldots p_j p_{j+1}$. If p_{j+1} <> p_{i+1}, set j = f(j) and repeat the previous step. Continue in this manner until a given $p_{j+1} = p_{i+1}$ or j = 0. Example 1, page 50, contains a pseudocode description of the algorithm.

Modifications to the KMP Algorithm

I wanted to use wildcard characters in my findcmd program, and because I could assume a fixed-length subject string (the length of DOS file names), I dispensed with the failure states. (In applications in which the subject strings are lengthy, failure states greatly increase the efficiency of the algorithm and should be implemented.) I also needed to use a backtracking facility to implement the asterisk operator—I will explain more about this when I discuss the program itself.

I also took the liberty of changing the interpretation of the wildcard characters and the DOS notion of the dot (.) file name extension. As in DOS, a question mark (?) in a pattern positionally matches any one character in a subject string. The asterisk (*), however, functions as a true regular-expression operator (à la Unix), matching zero or more characters. For example, the pattern *m*a*x* matches *max*, *maax*, and *mxaxx*, whereas the pattern *m?a?x* matches only *mxaxx*. Note that you can use more than one asterisk in a pattern (as long as they are not juxtaposed). Finally, a dot (.) in a file name is not treated specially; it is handled in the same manner as is any other valid file name character.

Implementation

I have divided the source code for findcmd into three modules, both as an aid to presentation and to simplify development. The file main.c (Listing One, page 92) contains the driving code of the program. It loops through each pattern (parameter) supplied on the command line; compiles that pattern into a state machine; then steps through each component of the path, comparing the compiled pattern with every file contained in that directory.

Step	Input	Curr State	Trans State	Notes
0	–	0	–	Initial
1	a	0	1	
2	a	1	2	
3	b	2	3	
4	b	3	4	
5	a	4	1	Failure
6	a	1	2	
7	b	2	3	
8	b	3	4	
9	c	4	5	
10	c	5	6	
11	–	H	–	Accept

Figure 4: *The sequence of transitions given a pattern P=aabbcc and an input I=aabbaabbcc*

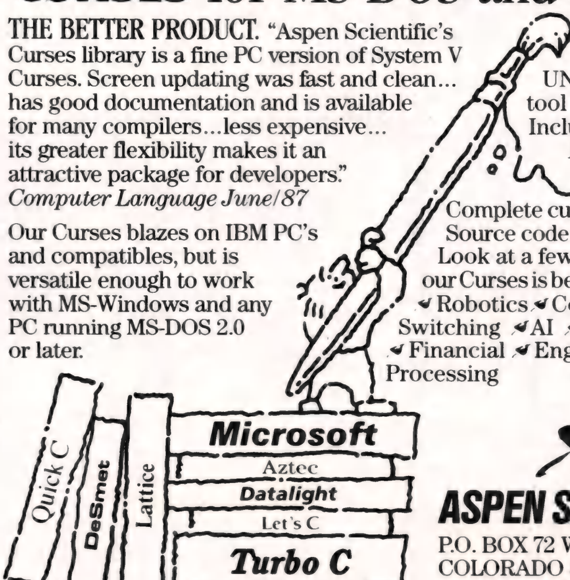
UNIX/C WINDOW DEVELOPMENT COMPATIBILITY with **CURSES** for MS-DOS and MS-OS/2.

THE BETTER PRODUCT. "Aspen Scientific's Curses library is a fine PC version of System V Curses. Screen updating was fast and clean... has good documentation and is available for many compilers...less expensive... its greater flexibility makes it an attractive package for developers." *Computer Language June/87*

Our Curses blazes on IBM PC's and compatibles, but is versatile enough to work with MS-Windows and any PC running MS-DOS 2.0 or later.

FREE-FAST UNIX compatible forms tool kit with source code. Included if you *ORDER RIGHT NOW.*

Complete curses tool kit: $119.
Source code available for: $289.
Look at a few of the applications our Curses is benefiting: Aerospace, Robotics, Consulting, Telephone Switching, AI, Voice Recognition, Financial, Engineering, Word Processing



ASPEN SCIENTIFIC
P.O. BOX 72 WHEAT RIDGE, COLORADO 80034-0072
(303) 423-8088

Trademarks: MS-DOS, MS-Windows, MS-OS/2, XENIX, Quick C (Microsoft); Unix (AT&T Bell Labs); Turbo C (Borland); Aztec (Manx); Lattice (Lattice); Let's C (Mark Williams); DeSmet (DeSmet Software); Datalight (Datalight).

CIRCLE 360 ON READER SERVICE CARD

(continued from page 49)

It then displays every file that successfully matched the pattern string in a manner analogous to Unix's *ls −l* command.

There are two important points to note. The program inserts the current directory (.) at the beginning of the path string to mimic the order in which DOS searches for commands. You should omit this in Unix ports of the program. The program also tests for a trailing backslash in several places in the code. This is largely for aesthetic reasons but it also mitigates DOS' tendency to "cough" when presented with double backslashes in its directory search functions.

Main.c also contains several ancillary routines. The function *nextdir( )* parses each directory segment of the *path* variable into individual strings; *putfile( )* prints, in a formatted manner, all the pertinent information about each successfully matched file; and *fdosdte( )* converts an internal DOS date into a formatted ASCII string.

The second module, dos.c (Listing Two, page 94), contains the two DOS-dependent functions *firstf( )* and *nextf( )*, which are used to access DOS Find First (*4EH*) and Find Next (*4FH*) system calls. In addition, *firstf( )* sets the disk transfer address to point to a C structure using the DOS system call Set Disk Transfer Address (*1AH*). In order to ensure the validity of the pattern matching operation, you need to gain access to every file in a directory. You therefore need to set the DOS search pattern to *.* (match all) and set the *CX* register to request all types of files (system, hidden, and so on). This module contains the only compiler-dependent piece of code in the program.

The third file, state.c (Listing Three) contains the code for the routines *inits( )* and *state( )*. The first function, *inits( )*, compiles the pattern strings into a state-machine format. It functions in a straightforward manner and is simple to understand. The other function, *state( )*, is slightly more subtle. The basic operation of the routine is simple: while in each state, it compares the search string with the pattern and, if they are equal, effects a transition to the next state; if the search string and pattern are not equal, it halts and returns an indicative value. If it makes transitions as far as the end state and exhausts the search string, it returns a value indicating a match.

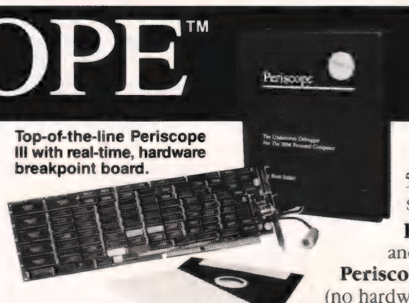
This procedure works quite well for patterns that do not contain any metacharacters (regular-expression operators). The addition of the question mark (*?*) operator (match any one character) is also straightforward because it really functions as a placeholder. It is only with the asterisk (*) operator (match zero or more repetitions of any character) that a problem arises.

Consider the pattern *m*ax*. The basic algorithm would be able to match this pattern successfully with the search strings *max* or *mmrax*. But what would happen with the search string *maaaax?* If you are not careful,

```
procfindfail()
begin
    f[0] := 0;
    for( j := 2 to N )
    do
        i := f[ j-1 ];
        while( p[j] <> p[i+1] AND i > 0 )
        do
            i := f[i];
        end while;
        if( p[j] <> p[i+1] AND i = 0 )
        then
            f[j] := 0;
        else
            f[j] := i + 1;
        end if;
    end for;
end proc;
```

***Example 1.** Pseudocode description of the failure*

PERISCOPE™ POWER



...Keeps you going full steam ahead when other debuggers let you down. With four models to pick from, you'll find a Periscope that has just the power you need.

. . . Start with the model that fits your current needs. If you need more horsepower, upgrade for the difference in price plus $10! And don't worry about having a lot more to learn . . . Even when you move to the most powerful model, Periscope III, an extra dozen commands are all that's involved.

Periscope's software is solid, comprehensive, and flexible. It helps you debug just about any kind of program you can write . . . thoroughly and efficiently. Periscope's hardware adds the power to solve the really tough debugging problems.

Periscope requires an IBM PC, XT, AT, or close compatible (Periscope III requires hardware as well as software compatibility); DOS 2.0 or later; 64K available memory; one disk drive; an 80-column monitor.

Periscope I includes a half-length board with 56K of write-protected RAM; break-out switch; software and manual for $345.

Periscope II includes break-out switch; software and manual for $175.

Periscope II-X includes software and manual (no hardware) for $145.

Periscope III includes a full-length board with 64K of write-protected RAM, hardware breakpoints and real-time trace buffer; break-out switch; software and manual. Periscope III for machines running up to 8 MHz is $995; for machines running up to 10 MHz, $1095.

Call Toll-Free for free information or to order your Periscope today!

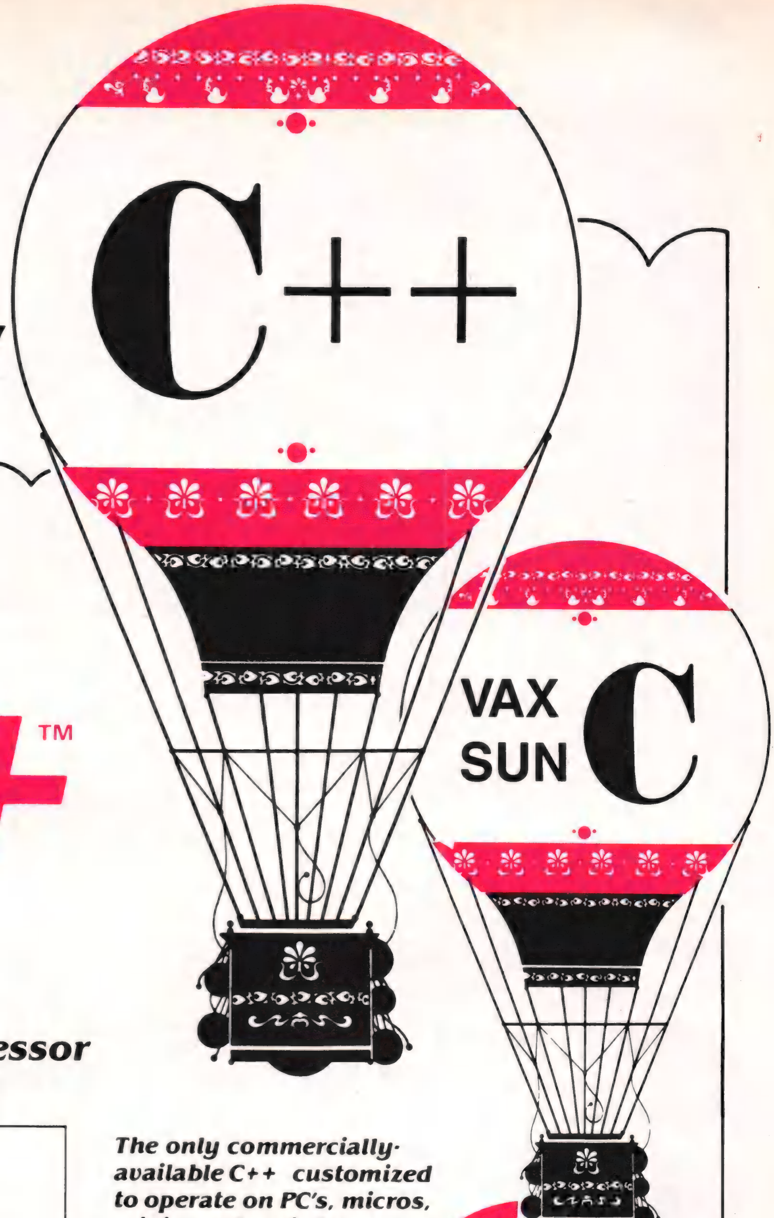
MAJOR CREDIT CARDS ACCEPTED. **800-722-7006**

The PERISCOPE Company, Inc.

14 BONNIE LANE
ATLANTA, GA 30328
404/256-3860

CIRCLE 214 ON READER SERVICE CARD

Take your favorite C compiler to new heights!



Designer C++™

- **Works with your present C Compiler**
  - **Object-oriented pre-processor**
    - **Strong type-checking**

BENEFITS:

- Incrementally add C++ features to C (switch-selectable)
- Makes C more suitable for
  — very large programs
  — more sophisticated applications
- Works with Sun's *dbxtool*
- Works with the C Compiler you now use
- More reusable code
- Resilient and bug-free code

FEATURES:

- Fully compatible with AT&T C++ standard
- Switch-selectable strong type checking
- Data abstraction
- Overloading of function names and operators
- Dynamic typing (virtual functions)
- User-defined implicit type conversion

The only commercially-available C++ customized to operate on PC's, micros, minis, and mainframes with popular C compilers, including:

VAX C
ULTRIX C
SUN C
MICROSOFT*
LATTICE*
GREEN HILLS
APOLLO
XENIX
HP-9000
UNISOFT

*Lattice and Microsoft versions of Designer C++ are known as Advantage C++

Oasys — A DIVISION OF XEL

60 Aberdeen Ave., Cambridge, MA 02138 (617) 491-4180
1219 Morningside Drive, Manhattan Beach, CA 90266 (213) 546-5814 (CA only)

Designer C++ is a joint trademark of XEL, Inc and Glockenspiel, Ltd. of Dublin. Ada is a trademark of the U.S. Government (AJPO). Advantage C++ is a trademark of Lifeboat Associates, Inc. Other trademarks are acknowledged to DEC, Lattice, Microsoft & Sun Microsystems, Inc

CIRCLE 254 ON READER SERVICE CARD

REAL UNIX. REAL FAST.

Today, there's only one place to get a real UNIX® System V Release 3 for your 80386 PC. From Microport, the company that, according to independent benchmarks, already brings you *the* fastest 286 UNIX system. Our System V/386™ is the real UNIX, developed by AT&T and Intel. As enhanced and extended by Microport, it can push every sinew of silicon in your 386 to its absolute limits.

And deliver almost unlimited speed and power in the process. Get true, multi-user multi-tasking performance today, not someday. For $199.00. Really.

Call quick for more info and a UNIX discount coupon.
(800) 722-UNIX/(800) 822-UNIX in CA

Real UNIX, $199

MICROPORT

UNIX is a registered trademark of AT&T. © 1987 Microport Systems, Inc.

CIRCLE 207 ON READER SERVICE CARD

The Advanced Programmer's Editor That Doesn't Waste Your Time

For DOS, Microport UNIX, or . . .

OS/2 Protected Mode

EPSILON

- Fast, EMACS-style commands—completely reconfigurable
- Run other programs without stopping Epsilon—concurrently!
- C Language support—fix errors while your compiler runs
- Powerful extension language
- Great on-line help system
- Multiple windows, files
- Regular Expression search
- Unlimited file size, line length
- Supports large displays
- 30 day money-back guarantee
- Not copy protected

Only $195

Lugaru Software Ltd.
5740 Darlington Road
Pittsburgh, PA 15217

Call (412) 421-5911
for IBM PC/XT/AT's or compatibles

CIRCLE 135 ON READER SERVICE CARD

PATTERN MATCHING
(continued from page 50)

state( ) will return a "no-match" indication when it compares *maa* with *max*. If you provide the function with a backtracking capability, however, when it encounters a no-match condition, it will be able to restore the environment to a previously saved state (actually one subject string character beyond the saved position) and resume the search.

I implemented the backtracking feature using a stack that lets the function store (and recall) state information for patterns containing one or more asterisks. Each time it encounters the wildcard operator (*case '*':*), *state( )* saves (pushes) both the current state and a pointer into the subject string onto the stack. Then, if it should encounter a no-match condition (*default:*), it can check the contents of the stack, and if it is nonempty, restore (pop) a saved state and continue processing.

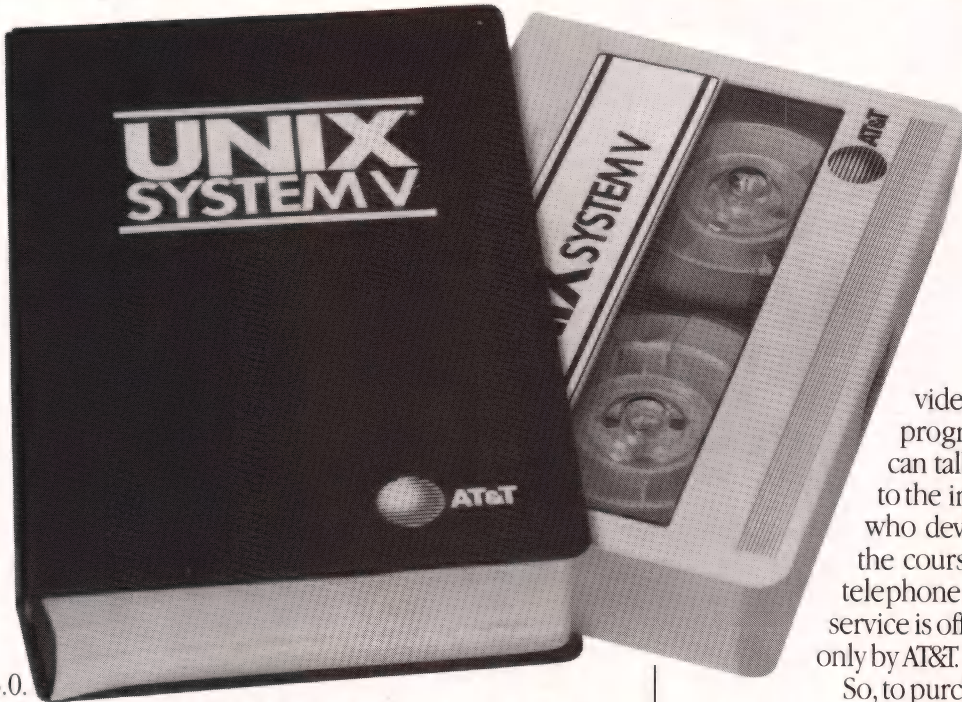
The file findcmd.h (Listing Four, page 107) contains all the global definitions, declarations, and macros referenced in the other three files.

Invoking the program is simple—just type *findcmd* followed by a list of patterns (files) for which you want it to search. Patterns can be as simple or complex as you wish. For example, if you want to find the directory that contains your favorite editor, just type *findcmd ed.exe*, and the program will search each directory component of your *path* variable for the file ed.exe. If you cannot remember whether the file is in .com or .exe format, type *findcmd ed.**, *findcmd ed**, or *findcmd ed????*. The latter two examples highlight the fact that findcmd, unlike COMMAND.COM, treats the period as an ordinary character in the file name. That is, a dot does not delimit the scope of the asterisk operator.

Portability

The program was written under MS-DOS using Version 3.0 of the Mark Williams C compiler. If you use another compiler, the only nonportable code is contained in the file dos.c. This file should not be too difficult to deal with, however. Most popular C compilers have a method of accessing DOS system facilities—just modify the two

Now there's a new UNIX® System video training program, from the people who wrote the book.



AT&T, the inventor of the UNIX System, now offers the most comprehensive and most current UNIX System training, even including UNIX System V Release 3.0. A complete curriculum, on videotape, through the new AT&T Videotape Library.

Modular Courses.

☐ Fundamentals of the UNIX System:
Basic
Intermediate
Advanced

☐ Shell Command Language for Programmers

☐ 'C' Language for Programmers

Our courses are created by AT&T instructors with 10 years of experience in grooming AT&T's own UNIX System professionals. Courses are modular and can be used in entirety or in sections.

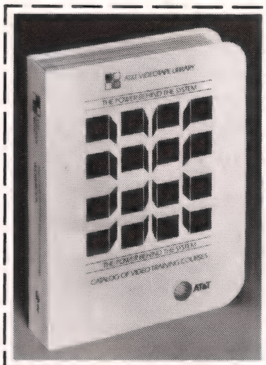
More than a 'taped lecture.'
Each course blends professionally developed graphics and text into a high-interest, easy-to-follow learning experience. And each course is self-contained, including sample programs and workbook exercises for thorough retention of skills.

Questions? Call us.
If you have any technical questions after you have reviewed your videotape program, you can talk directly to the instructors who developed the courses. This telephone support service is offered only by AT&T.

So, to purchase or lease a video training program that is authoritative, current, and complete, call or write now for more information on the AT&T Videotape Library. A demonstration video is available.

© 1987 AT&T

Call 1 800 247-1212, Ext. 1001, or mail this coupon.



AT&T Videotape Library, *Dr.D 10/87*
P.O. Box 2160, Jacksonville, FL 32232-9912

☐ Please rush me your course catalog on the AT&T Videotape Library for Computer Systems Training.
☐ Please send me information on your classroom training programs.
☐ UNIX System training ☐ Business applications training
☐ Data communications and networking training

Name ______
Title ______
Company ______
Address ______
City ______ State ______ Zip ______
Phone ( )

CIRCLE 396 ON READER SERVICE CARD

AT&T
The right choice.

VEDIT PLUS

#1 PROGRAMMABLE EDITOR

Call 1-800-45-VEDIT for FREE Fully Functional Demo Disk

Stunning speed. Unmatched performance. Total flexibility. Simple and intuitive operation. The newest VEDIT PLUS easily satisfies the most demanding computer professional.

Try a Dazzling Demo Yourself.

The free demo disk is fully functional—you can try all features yourself. Best, the demo includes a dazzling menu-driven tutorial—you experiment in one window while another gives instructions.

The powerful "macro" programming language helps you eliminate repetitive editing tasks. The impressive demo/tutorial is written entirely as a "macro"—it shows that no other editor's "macro" language even comes close. And VEDIT PLUS is only 40K in size.

Go ahead. Call for your free demo today, You'll see why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

Only VEDIT PLUS is this Flexible.

The installation lets you pick from closely emulating the keyboard layout of Word Perfect, WordStar and others. Or you can easily create your own layout and even your own editing functions. Supports any screen size—you pick screen colors and attributes.

Supports the IBM PC, XT, AT and PS/2. Also supports MultiLink, PC-MOS/386, Concurrent DOS and most networks. Also available for MS-DOS, FlexOS (protected mode), CP/M-86 and CP/M. (Yes, we support windows on most CRT terminals, including CRTs connected to an IBM PC.) Order direct or from your dealer. $185.

Special: VEDIT (single file, no windows) for CP/M—$49.

- **Fully Network Compatible**
- **Call for XENIX-286 version**
- **30 Day Money-back guarantee**

Compare Features and Speed

	BRIEF	Norton Editor	PMATE	VEDIT PLUS
'Off the cuff' macros	No	No	Yes	Yes
Built-in macros	Yes	No	Yes	Yes
Keystroke macros	Only 1	No	No	Unlimited
Multiple file editing	20+	2	No	20+
Windows	20+	2	No	20+
Macro execution window	No	No	No	Yes
Pop-up menus	No	No	No	Yes
Execute DOS commands	Yes	Yes	Yes	Yes
Automatic processing of Compiler errors	Yes	No	No	Yes
"Cut and paste" buffers	1	1	1	36
Undo line changes	Yes	No	No	Yes
Paragraph justification	No	No	No	Yes
Convert to/from WordStar	No	No	No	Yes
On-line calculator	No	No	No	Yes
Configurable Keyboard	Hard	No	Hard	Easy
43 line EGA support	Yes	No	No	Yes
Manual size/index	250/No	42/no	469/Yes	380/Yes
Benchmarks in 120K File:				
2000 replacements	1:15 min	34 sec	1:07 min	6 sec
Pattern matching search	20 sec	Cannot	Cannot	2 sec
Pattern matching replace	2:40 min	Cannot	Cannot	11 sec

VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. PMATE is a trademark of Phoenix Technologies Ltd. Norton Editor is a trademark of Peter Norton Computing Inc. MultiLink and PC-MOS/386 are trademarks of the The Software Link, Inc. CP/M and FlexOS are trademarks of Digital Research. MS-DOS is a trademark of Microsoft.

*Also available for TI Professional, Tandy 2000, DEC Rainbow, Wyse WY700 and others.
*Demo disk is fully functional, but does not readily write large files.

CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103
(313) 996-1299, TELEX 701821

CIRCLE 122 ON READER SERVICE CARD

PATTERN MATCHING
(continued from page 52)

C functions to work within your compilation environment.

For those who wish to port the program to a Unix system, the same caveat applies: just modify the functions in dos.c. You will have to completely rewrite the routines but, again, this should not be difficult. A good description of how to access a Unix directory is in Brian Kernighan and Rob Pike's *The Unix Programming Environment* (see bibliography).

Extensions

This program can accommodate several extensions. One that I like (I have implemented it on both my PC and Unix versions) is the addition of a switch on the command line to direct the program to access an alternate environment variable in lieu of *path*. This is useful if you tend to keep application source files scattered all over your disk. You could set this alternate variable (in the same format as you would your *path*) to point to all your source code directories. Then, by setting one switch on the command line, findcmd can search an alternate set of directories for you.

You can modify the functions contained in state.c for independent use by incorporating the routine *inits( )* into a separate program. This program's only function would be to compile pattern strings and store the results in a file. Then, at execution time, a program performing the actual comparisons would not incur the expense of "compiling" the patterns—it would just read them from disk. This would operate in a manner similar to the Unix utilities *regcomp( )* and *regex( )*.

Additional regular-expression operators could be incorporated into the algorithm to add more power and flexibility to search patterns. I have found, however, that the operators now included in the program are adequate for everyday use (and there are better algorithms than KMP for recognizing complex regular expressions).

Finally, consider how simple the changes would be to transform this program into an *ls* command (as if you needed another one!).

Conclusions

In any development environment, no single programming tool is sufficient to satisfy every need. Any instrument or methodology that can spare us from the tedium of a routine chore, or enable us to become more productive, should be embraced with open arms. State machines are just such a tool and can benefit all programmers.

Availability

All the source code for articles in this issue is available on a single disk. To order, send $14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call (415) 366-3600, ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro). If you would rather not have to retype the entire program send me a check for $6 and I will mail you an MS-DOS floppy (360K format) containing the program and source code.

Bibliography

Aho, Alfred V; Hopcroft, John E.; and

Beyond the 64K Boundary
With Automatic Overlays

64180/Z80

Using bank switched or disk based overlays, your modular program can be as **large as you want.** *Our linker generates the code to switch in the correct bank at the right time, or load the right module from disk. Includes source to the overlay loader for easy customization to your requirements.* **SLRNK Plus 3.0,** *from the leaders in 8-bit development tools.*

$195

SLR Systems
1622 N. Main St.
Butler, PA 16001
(412) 282-0864 (800) 833-3061 TELEX: 559215

CIRCLE 78 ON READER SERVICE CARD

8031
FORTH DEVELOPMENT ENVIRONMENT

Take advantage of Bryte's tools to make your job easier:

- Bryte's development environment uses BRYTE-FORTH on the actual production hardware during product development. No emulators, no changes, no surprises.
- Optional PC-based cross-development tools use DOS files as microcontroller mass storage. These files can be used to generate compact EPROM images, detailed listings, and cross-references.

Why not start developing the Bryte way today?

BRYTE-FORTH 8Ø31 EPROM (includes 13Ø page User's Manual)	100.00
Utility disk(s)	65.00*
Cross-compiler/Cross-assembler	235.00*
8Ø31 unlimitted quan. license	1000.00*

* Includes complete source code

bryte computers, inc.
P.O. Box 46
Augusta, ME 04330-0046
207/547-3218

CIRCLE 387 ON READER SERVICE CARD

THE CURE FOR COMMON CODE

Are you getting the recommended monthly allowance of C, Assembly, Forth, Pascal, BASIC or Modula-2? Subscribe to *Dr. Dobb's Journal of Software Tools* and you won't catch any nasty bugs again!

Each month the Doctor brings you aid for ailing algorithms and the cure for common code. For the latest developments in software design and pages of code that will make you a more productive programmer, take the Dr. Dobb's prescription.

For more than a decade, the programming elite have known *Dr. Dobb's Journal* to be the foremost source of software tools. Subscribe now and get your monthly dose from the Doctor.

Dr. Dobb's Journal of Software Tools

The R$_x$ for Programmers

Subscribe Now & Save Over 15% Off the Newsstand Price!

SUBSCRIBE AND SAVE!

$5 SAVINGS

Subscribe to

DR. DOBB'S JOURNAL OF SOFTWARE TOOLS

and save over $5—a 15% savings off the cover price!

☐ Please charge my: ☐ Visa ☐ Master Card ☐ American Express

☐ Payment enclosed ☐ Bill me later

Card # ______ Exp. date ______

Signature ______

Name ______

Address ______

City ______ State ______ Zip ______

ONLY $29.97! YOU SAVE OVER $5.00

Savings based on a full one-year cover price of $35.40. Canada & Mexico add $10 for surface mail per year. All other countries add $27 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc. **3459**

SUBSCRIBE AND SAVE!

$5 SAVINGS

Subscribe to

DR. DOBB'S JOURNAL OF SOFTWARE TOOLS

and save over $5—a 15% savings off the cover price!

☐ Please charge my: ☐ Visa ☐ Master Card ☐ American Express

☐ Payment enclosed ☐ Bill me later

Card # ______ Exp. date ______

Signature ______

Name ______

Address ______

City ______ State ______ Zip ______

ONLY $29.97! YOU SAVE OVER $5.00

Savings based on a full one-year cover price of $35.40. Canada & Mexico add $10 for surface mail per year. All other countries add $27 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc. **3459**

COMMENTS & SUGGESTIONS

Dear Reader, **October 1987, #132**

Dr. Dobb's has a long tradition of listening to its readers. We like to hear when something really helps or, for that matter, bothers you. In this hectic world of ours, however, it is often difficult to take time to write a letter. This card provides you with an easy way to correspond and, if you include your name and address, we may use appropriate comments in The Letters column. Simply fill it out and drop it in the mail.

—Ed

Which articles or departments did you enjoy the most this month? Why?

______

Comments or suggestions ______

______

Name: ______

Address: ______

No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of
Software Tools
Box 3713
Escondidio, CA 92025-9843

No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of
Software Tools
Box 3713
Escondidio, CA 92025-9843

PLACE
STAMP
HERE

Dr. Dobb's Journal of
Software Tools
501 Galveston Drive
Redwood City, CA 94063

Dr. Dobb's Journal of Software Tools

The Rx for Programmers

Subscribe Now & Save Over 15% Off the Newsstand Price!

PATTERN MATCHING
(continued from page 55)

Ullman, Jeffrey D. *The Design and Analysis of Computer Algorithms.* Reading, Mass.: Addison-Wesley, 1974.

Aho, Alfred V; and Ullman, Jeffrey D. *Principles of Compiler Design.* Reading, Mass.: Addison-Wesley, 1979.

Baase, Sarah. *Computer Algorithms.* Reading, Mass.: Addison-Wesley, 1979.

Barrett, William A.; and Couch, John D. *Compiler Construction: Theory and Practice.* Chicago, Ill.: Science Research Associates Inc., 1979.

Kernighan, Brian; and Pike, Rob. *The Unix Programming Environment.* Englewood Cliffs, N.J.: Prentice-Hall, 1984: 208–219.

DDJ

(Listings begin on page 92.)

Vote for your favorite feature/article.
Circle Reader Service **No. 4.**

Brand New From Peter Norton A PROGRAMMER'S EDITOR

that's *lightning fast* with the *hot* features programmers need

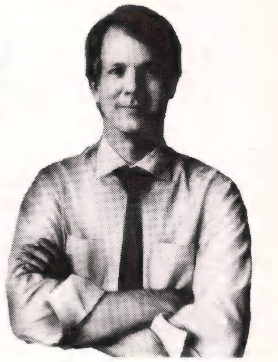
only $50

THE NORTON EDITOR™

Direct from the man who gave you *The Norton Utilities, Inside the IBM PC,* and the *Peter Norton Programmer's Guide.*

"This is the programmer's editor that I wished I'd had when I wrote my *Norton Utilities.* You can ***program your way to glory*** with *The Norton Editor.*"

Peter Norton



***Easily customized,* and saved**
Split-screen editing
A wonderful condensed/outline display
***Great for* assembler, Pascal *and* C**

Peter Norton Computing, Inc., 2210 Wilshire Boulevard, Santa Monica, CA 90403, 213-453-2361. Visa, Mastercard and phone orders welcome.

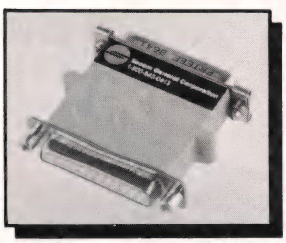
CIRCLE 243 ON READER SERVICE CARD

Would you like copy protection and customer satisfaction?

There's a better way to protect your software. It's called the Secom Key, and it works.

- *The Key is completely transparent to the end user.*
- *Won't interfere with peripheral operations.*
- *Doesn't occupy the disk drive.*
- *The Key allows unlimited backup copies.*
- *Makes site licensing easy and auditable.*
- *Easily installed. Uses only 1000 bytes.*
- *Over 60,000 have been sold worldwide.*
- *Same size as RS-232 plug.*
- *Available in quantities for as low as $19.95.*

For more information, contact Secom Information Products Co.



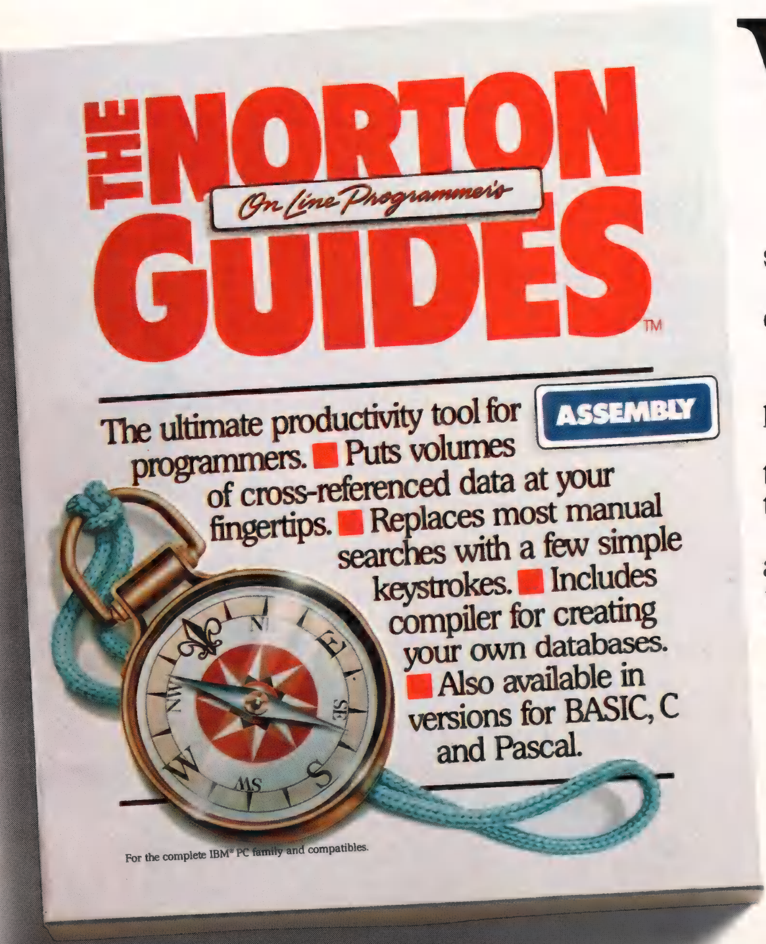
500 Franklin Square
1829 East Franklin Street
Chapel Hill, NC 27707

The Secom Key... for real software protection.

Secom Information Products Company
A Subsidiary of Secom General Corporation
Call Toll-free 1-800-843-0413

CIRCLE 394 ON READER SERVICE CARD

Peter Norton new programmi who hate



Nobody ever said programming PCs was supposed to be easy.

But does it have to be tedious and time-consuming, too?

Not any more.

Not since the arrival of the remarkable new program on the left.

Which is designed to save you most of the time you're currently spending searching through the books and manuals on the shelf above.

The Norton On-Line Programmer's Guides™ are a quartet of pop-up reference packages that do the same things in four different languages.

Each package consists of two parts: A memory-resident Instant Access™ program. And a comprehensive, cross-referenced database crammed with just about everything you need to know to program in your favorite language.

And when we say everything, we mean everything.

Everything from information about language

Designed for IBM® PC, PC-AT and DOS compatibles. Available at most software

announces a
ng tool for people
e manual labor.

syntax to a variety of tables, including ASCII characters, line drawing characters, error messages, memory usage maps, important data structures and more.

How much more?

Well, the databases for BASIC, C and Pascal give you detailed listings of all built-in and library functions.

While the Assembly database delivers a complete collection of DOS service calls, interrupts and ROM BIOS routines.

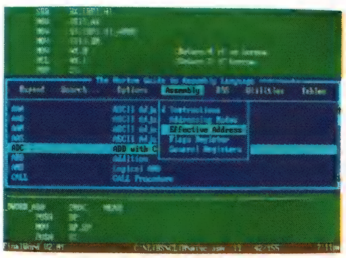
You can, of course, find most of this information in the books and manuals on our shelf.

But Peter Norton—who's written a few books himself—figured you'd rather have it on your screen.

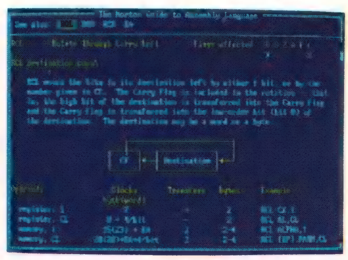
In seconds.

In full-screen or moveable half-screen mode.

Popping up right next to your work. Right where you need it.



A Guides reference summary screen (shown in blue) pops up on top of the program you're working on (shown in green).



Summary data expands on command into extensive detail. And you can select from a wide variety of information.

This, you're probably thinking, is precisely the kind of thinking that produced the classic Norton Utilities.™

And you're right.

But even Peter Norton can't think of everything.

Which is why there's a built-in compiler for creating databases of your own.

And why all Guides databases are compatible with the Instant Access program in your original package.

So you can add more languages without spending a lot more money.

To get more information, call your dealer. Or call Peter Norton at 213-453-2361.

And ask for some guidance.

Peter Norton
COMPUTING

dealers, or direct from Peter Norton Computing, Inc., 2210 Wilshire Blvd., #186, Santa Monica, CA 90403. 213-453-2361, Fax 213-453-6398, MCI Mail: PNCI © 1987 Peter Norton Computing

CIRCLE 87 ON READER SERVICE CARD

Listing One *(Text begins on page 18.)*

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
*   THIS FILE CONTAINS EXCERPTS FROM THE APPLETALK SOURCES,
*        VERSION 39, AUGUST 1985, AS MODIFIED BY DARTMOUTH COLLEGE
*        TO PRODUCE THE ASYNC APPLETALK DRIVER (.BPP) VERSION 1.2
*        (ASYNC APPLETALK INSTALLER VERSION 2.1) OF MAY 1987.
*
*        THESE EXCERPTS CONTAIN INFORMATION OF TWO TYPES:
*                 1) CODE WRITTEN ENTIRELY AT DARTMOUTH COLLEGE;
*                 2) CODE WHICH IS FUNDAMENTALLY SIMILAR TO THE
*                          PRELIMINARY APPLETALK SOFTWARE DISTRIBUTED
*                          WITHOUT RESTRICTION AT THE APPLEBUS DEVELOPER'S
*                          CONFERENCE IN CUPERTINO, CA IN MAY, 1984.
*
*        PORTIONS OF THIS CODE ARE COPYRIGHT OF THE TRUSTEES OF
*        DARTMOUTH COLLEGE OR APPLE COMPUTER INC.
*
*        THESE CODE SEGMENTS ARE PROVIDED FOR INFORMATION ONLY.  NO
*        GUARANTEE OF CORRECT OPERATION IS PROVIDED.
*
*        FOR MORE INFORMATION ABOUT THIS CODE, CONTACT:
*
*        Rich Brown
*        Manager of Special Projects
*        Dartmouth College
*        Kiewit Computer Center
*        Hanover, NH 03755
*        603/646-3648
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

***** file ALAPDEFS.A *****

; AALAPdefs.a contains all the special definitions which were not
; needed for .MPP .
;
; .BPP et al should now use unmodified versions of:
;                 {AIncludes}atalkequ.a
;                 lapdefs.a
;                 vardefs.a
;
; Created 31 Mar 87   reb
;
; AALAP constant defs
;
MaxLAPFrmLen EQU    586+13+3+2          ; DDP data + DDP hdr + LAP hdr + CRC
FrameChar    EQU    $A5                 ; the Framing Char
qFrmChar     EQU    -91                 ; for moveq instructions
DLE          EQU    $10
Xoff         EQU    $13
Xon          EQU    $11

lapIM        EQU    $86                 ; I aM
lapUR        EQU    $87                 ; yoU aRe (sorry for these names...)
qlapIM       EQU    -122
qlapUR       EQU    -121

noansalrt    EQU    -15998
portncalrt   EQU    -15997

; Added constant return value for AALAP --
noAnswer          EQU     -95          ; same as excessCollsns in real AtalkEqu

AAOfst            EQU     10000      ; 0 or 10000 (for final version)

;+ MPP (Status calls to NBP, DDP and AALAP)

GetStats          EQU     400        ; (ABLAP) get the statistics
GetMyName         EQU     AAOfst+255 ; get the name of the ATalk driver
(AALAP)
GetChar           EQU     AAOfst+254 ; get the most recently received char
(AALAP)
GetLAPStatus      EQU     AAOfst+253 ; return AALAP status  (AALAP)

;+ MPP (Control calls to NBP, DDP, and AALAP)

FirstAPP          EQU     AAOfst+237 ; First APP control call
DoWarnings        EQU     AAOfst+237 ; Put up the specified alerts
(AALAP)
PutChar           EQU     AAOfst+238 ; Loop 'til TBMT, then output the char
(AALAP)
```

(continued on page 62)

The Software Family

Dept DD107, 649 Mission Street
San Francisco, CA 94105

(1)800-443-7176

In California or outside U.S.

(415)583-4166

Basic **C**
Turbo Pascal **dBase**
Publishing **AI**

Trust TSF to provide the best value and service!

* Technical advice and support by programmers
* Honest and equitable business practices
* Prompt, flexible service to meet your needs

We accept checks, Visa, MasterCard, American Express and COD. We charge your card only when we ship and do **not** add a surcharge. We provide free UPS delivery on software orders over $100 ($3 delivery charge on orders under $100). We add actual charges for UPS Blue Label or Federal Express rush service. Our COD fee is $2. We allow return privileges on most products. We carry a large inventory and ship promptly, so you receive your shipment within 3 to 6 working days of placing your order. Give us a try!

October Specials

Microsoft **Quick C** (list $99) **$62**

Blaise **Turbo C Tools** (list $129) **$95**
ISR/TSR support, critical error trapping, direct video access, more.

Matrix **Synergy Toolkit** ($395) **$289**
Complete user interface toolkit including multi-tasking, windows, clipboard data transfer, menus, icons, graphics, mouse/cursor control and code generator for C and Turbo Pascal. Includes bindings for most popular Basic, C and Pascal compilers.

Metagraphics **Turbo Window** ($95) **$75**
Windows, graphics, multiple fonts (monospace and proportional) & mouse/cursor control. Specify Turbo C or Turbo Pascal.

Softfocus **Btree & ISAM** (list $115) **$99**
Complete multi-key ISAM file system with source.

Prices good until November 15, 1987

Aldebaran **Source Print** (list $75) **$59**
Aldebaran **Tree Diagrammer** (list $55) **$42**
Blaise **C Tools +** (list$175) **$125**
Blaise **Turbo C Tools** (list $129) ...... **$95** *New!*
Blaise **Turbo Power Tools +** (list $99) **$79**
Borland **Turbo C** (list $99) **$64**
Creative **Vitamin C** Specify compiler (list $225) **$159**
Creative Programming **VC Screen** (list $100) **$79**
Digitalk **Smalltalk/V** (list $99) **$82**
Gimpel **PC-Lint** (list $139) **$103**
Greenleaf **Data Windows** Specify compiler ($225) **$157**
Guidelines **C++** for Microsoft C (list $195) **$175**
Logitech **Modula-2 Apprentice Package** ($99) **$79**
Logitech **Modula-2 Wizard's Package** ($199) **$159**
Matrix **Synergy Development Toolkit** (list $395) ...... **$289** *Save!*
Metagraphics **MetaWindow/All Language** ($195) **$159**
Metagraphics **TurboWindow/C** ($95) ...... **$75** *Save!*
Metagraphics **TurboWindow/Pascal** ($95) ...... **$75** *Save!*
MicroHelp **Mach2 for Microsoft Basic** ($69) **$55**
windows, fast i/o, critical error trapping, more
MicroHelp **Mach2 for Turbo Pascal** ($69) **$55**
Microsoft **C v5** w/Codeview & Quick C (list $450) ...... **$279** *New!*
Microsoft **Quick C** (list $99) ...... **$62** *Save!*
Microsoft **Fortran** w/Codeview (list $450) ...... **$270** *New!*
Microsoft **Quick Basic v3** (list $99) ...... **$45** *Save!*
You pay TSF $65 and get $20 rebate from Microsoft
MKS **Toolkit** Korn shell, vi, grep, 30 more ($139) **$115**
Periscope **Periscope I** (list $345) **$279**
Periscope **Periscope II** (list $175) **$139**
Periscope **Periscope III** (list $995) **$799**
Periscope **Periscope III 10Mhz** (list $1095) **$876**
Phoenix **PFix-86+** (list $395) **$250**
Phoenix **Plink-86+** (list $495) **$315**
PMI **Repertoire** for Logitech Modula-2 ($89) **$75**
Softfocus **Btree & ISAM** C DBMS w/source ($115) ...... **$99** *Save!*
Sterling Castle **Basic Development Tools** ($99) **$79**
Sterling Castle **Blackstar "C" Functions** ($129) ...... **$99** *New!*
Texas Instruments **PC Scheme** great AI intro **$77**
Turbo Power **Extender** (list $85) **$68**
Turbo Power **Optimizer** (list $75) **$59**
Turbo Power **Optimizer** w/Source (list $175) **$99**
Turbo Power **TDebug Plus v2** (list $60) **$48**
Turbo Power **Turbo Utilities** w/source ($95) **$76**
Unipress **Phact** RDBMS w/ report & query ($249) ...... **$199** *New!*

TSF carries hundreds of products from dozens of publishers. Call or write for a FREE catalog or a price quote.

CIRCLE 230 ON READER SERVICE CARD

Listing One *(Listing continued, text begins on page 18.)*

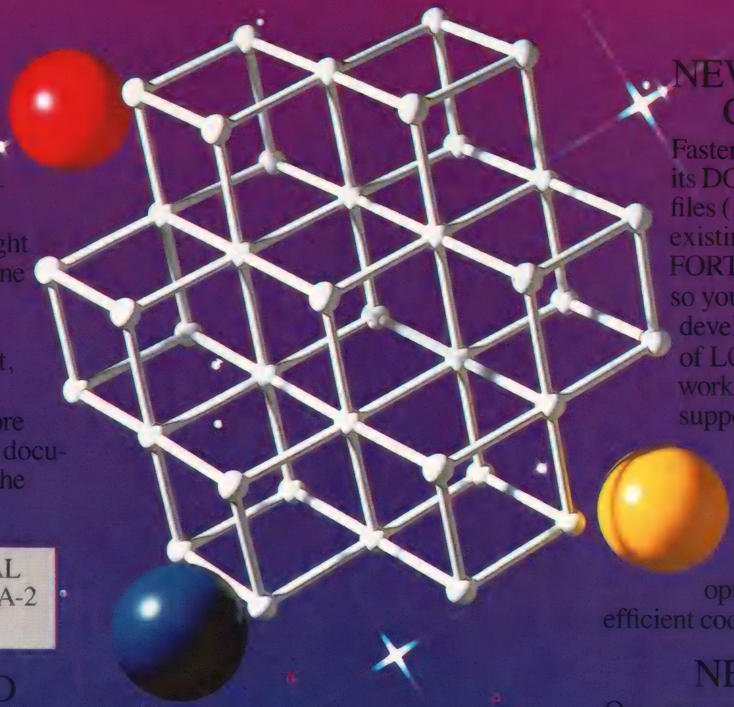
```
ReInitAALAP     EQU     AAOfst+239 ; ReInitialize the AALAP variables & SCC
(AALAP)
GetNNNN         EQU     AAOfst+240 ; Do NNNN using SysNetNum and sysLAPAddr
(AALAP)
SetBaud         EQU     AAOfst+241 ; Set the baud rate of the SCC
(AALAP)
LastAPP         EQU     AAOfst+241
          EJECT
;
; LAP variables
;
WDSPtr       EQU     MPPVarsEnd          ; (4) WDS pointer saved here on writes
LAPWrtRtn    EQU     WDSptr+4            ; (4) return adrs of LAPWrite caller
SaveA45      EQU     LAPWrtRtn+4         ; (8) A4 and A5 saved here on interrupt
SaveDskRtn   EQU     SaveA45+8           ; (4) DskRtnAdr saved here for
PollProc
SavePS       EQU     SaveDskRtn+4        ; (4) in AALAP, the real PollProc's
address
SaveBIn      EQU     SavePS+4            ; (4) .BIN DCE saved here (for close)
SaveBOut     EQU     SaveBIn+4           ; (4) .BOUT DCE saved here (for close)
SaveVects    EQU     SaveBOut+4          ; (16) SCC interrupt vectors saved
here
SaveRegs     EQU     SaveVects+16        ; (20) Registers saved here across
PollProc

;
; Variables for Lisa/Mac hardware differences
;
VAVBufA      EQU     SaveRegs+20         ; Pointer to VIA or a $FF word
STLth        EQU     6                   ; Size of STData area
VSTData      EQU     VAVBufA+4           ; Data string to SCC after send
VDisTxRTS    EQU     VSTData+1           ; This is the DisTxRTS byte
EndOrigStuff EQU     VSTData+STLth       ;

;
; AALAP varibles
;
tWDSptr      EQU     EndOrigStuff+2      ; (4) WDS ptr of frame being tx
qWDSptr      EQU     tWDSptr+4           ; (4) WDS of a queued DevMgr frame
LastXmit     EQU     qWDSptr+4           ; (4) Ticks at time of last char sent
LastRcv      EQU     LastXmit+4          ; (4) Ticks at time of last good
received frame
LAPStash     EQU     LastRcv+4           ; (4) Pointer to next received char's
place
LAPFetch     EQU     LAPStash+4          ; (4) Pointer to next char to xmit
LAPInBuf     EQU     LapFetch+4          ; (4) Pointer to the LAP input buffer
IMURwds      EQU     LAPInBUf+4          ; (8) WDS for IM or UR frames
BusyBuf      EQU     IMURwds+8           ; (16) Holds up to 16 chars rcvd while
doingRead
BusyStash    EQU     BusyBuf+16          ; (4) pointer to next space in BusyBuf
BusyFetch    EQU     BusyStash+4         ; (4) pointer to next char to remove
IMURbuf      EQU     BusyFetch+4         ; (8) Holds IM or UR (starting at odd
adrs)
InputCRC     EQU     IMURBuf+8           ; (2) CRC for the receiver
OutputCRC    EQU     InputCRC+2          ; (2) CRC for the transmit side
RcvdLen      EQU     OutputCRC+2         ; (2) Number of chars received
TxCount      EQU     RcvdLen+2           ; (2) Number of char's transmitted
CRCBuf       EQU     TxCount+2           ; (2) Two bytes for the CRC for
xmission
RandomSeed   EQU     CRCBuf+2            ; (2) Seed for random number generator
LastRxCh     EQU     RandomSeed+2        ; (2) Lsbyte is last rcvd char, else
$FFFF
AALAPbaud    EQU     LastRxCh+2          ; (2) Current baud rate of the LAP
SentChar     EQU     AALAPbaud+2         ; (1) True if TxNextCh sent a char
nFrmChr      EQU     SentChar+1          ; (1) True if we must send a FrameChar
nCRC         EQU     nFrmChr+1           ; (1) True if we must send the CRC
EscIn        EQU     nCRC+1              ; (1) Escaping flag for the receiver
EscOut       EQU     EscIn+1             ; (1) Transmitter is sending an escaped
char
RcvdXoff     EQU     EscOut+1            ; (1) We received Xoff
AALAPup      EQU     RcvdXoff+1          ; (1) true if we've handshook IM & UR
AALAPstuck   EQU     AALAPup+1           ; (1) true if we have NNNN conflict
InpState     EQU     AALAPstuck+1        ; (1) 0 = idle; <> 0 = in a frame
stillBusy    EQU     InpState+1          ; (1) true if still processing a read
nXon         EQU     stillBusy+1         ; (1) true if we sent Xoff
SendingIMUR  EQU     nXon+1              ; (1) true if sending AALAP control frame

 _AssumeEq    (InpState+1),stillBusy         ; tst.w InpState(A4) in
 _AssumeEq    (InpState**$FFFFFFFE),InpState ;   myPollProc fails otherwise

             IF      debug THEN          ; doing statistics
```

(continued on page 65)

SOFTWARE ENGINEERING COMES OF AGE.

ANNOUNCING LOGITECH MODULA-2 VERSION 3.0

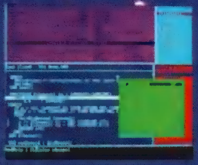
Modula-2 is the language of choice for modern software engineering, and LOGITECH Modula-2 is the most powerful implementation available for the PC. The right language and the right tools have come together in one superior product. Whether you're working on a small program or a complex project, with LOGITECH Modula-2 Version 3.0 you can write more reliable, maintainable, better documented code in a fraction of the time at a fraction of the cost.



FREE TURBO PASCAL TO LOGITECH MODULA-2 TRANSLATOR

NEW, IMPROVED DEBUGGERS

Time gained with a fast compiler can be lost at debug time without the right debugging tools. With the powerful Logitech Modula-2 Debuggers you can debug your code *fast*, and dramatically improve your overall project throughput. The Post Mortem Debugger analyzes the status of a program after it has terminated while the dynamic, Run Time Debugger monitors the execution of a program with user-defined break points. With their new, mouse based, multiple-window user interface these powerful debugging tools are a pleasure to use.



NEW, INTELLIGENT LINKER

Links only those routines from a particular module that you need, so you eliminate unreferenced routines and produce smaller, more compact executable files.

NEW, IMPROVED COMPILER

Faster and more flexible. Now its DOS linker compatible object files (.OBJ) can be linked with existing libraries in C, PASCAL, FORTRAN and ASSEMBLER — so you can build on previous development and put the power of LOGITECH Modula-2 to work for you right now. Fully supports Wirth's latest language definition, including LONGINT and LONGSET, which provides large set support including SET of CHAR. Provides optimization for tighter, more efficient code generation.

NEW EDITOR

Our new, mouse based editor is fully integrated, easy to learn, fast and easy to use, and very customizable. Its multiple, overlapping windows and color support make it easy to manage parts of one file or several files on the screen at one time. You'll love using it — with or without a mouse.

Call for information about our VAX/VMS version, Site License, University Discounts, Dealer & Distributor pricing.

To place an order call toll-free:
800-231-7717
In California:
800-552-8885

☐ LOGITECH Modula-2 V. 3.0 Compiler Pack — **$99**
Compiler in overlay and fully linked form, Linkable Library, Post Mortem Debugger, Point Editor

☐ LOGITECH Modula-2 V. 3.0 Toolkit — **$169**
Library sources, Linker, Run Time Debugger, MAKE, Decoder, Version, XRef, Formatter

☐ LOGITECH Modula-2 V. 3.0 Development System — **$249**
Compiler Pack plus Toolkit

☐ Turbo Pascal to Modula-2 Translator — **FREE**
With Compiler Pack or Development System

☐ Window Package — **$49**
Build true windowing into your Modula-2 code.

☐ Upgrade Package
Call LOGITECH for information or to receive an order form.

Add $6.50 for shipping and handling. California residents add applicable sales tax. Prices valid in U.S. only. Total Enclosed $______

☐ VISA ☐ MasterCard ☐ Check Enclosed

Card Number ______ Expiration Date ______

Signature ______

Name ______

Address ______

City ______ State ______

Zip ______ Phone ______

LOGITECH

LOGITECH, Inc.
6505 Kaiser Drive, Fremont, CA 94555
Tel: 415-795-8500

In Europe: LOGITECH, Switzerland
Tel: 41-21-87-9656 Telex 458 217 Tech Ch

In the United Kingdom: LOGITECH, U.K.
Tel: 44908-368071 Fax: 44908-71751

Turbo Pascal is a registered trademark of Borland International. VAX and VMS are registered trademarks of Digital Equipment Corp.

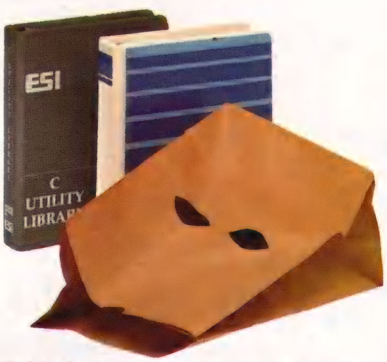
CIRCLE 257 ON READER SERVICE CARD

Some Of The Most Famous Faces In Software Use Our C Functions

Our famous customers are a little camera shy. It's not that they are embarrassed by being Essential C Utility Library users. They just don't want us shouting their names from the roof tops.

The prestige of our users is not the primary reason to buy the Essential C Utility Library. The increased speed, features, and size efficiency of our products are the factors that demand their use. Our library contains *over 400 functions,* all designed with elegance in mind.

However, should curiosity get the best of you, call us at 201-762-6965 and we'll drop a few highly impressive names on you.



Behind every great program is a great library.

What's a Library Without a Librarian?

Our library comes complete with a sophisticated source code librarian. Now you can maintain current versions and conserve disk space. We want your development work to go as smoothly as possible.

No Royalties, 30 Day Guarantee

If within 30 days you don't find our library totally satisfactory, bag the whole thing and receive a complete refund. There are no royalties associated with the library.

Functions At A Glance

- Fastest screen output available.
- Save/Restore color screens in 1/10 sec.
- Pop-up block cursor menus
- Save/Restore windows to disk or memory
- 50 functions for business graphics
- dozens of string formats
- time and date arithmetic
- julian and day-of-week
- Ctrl-Break key trapping
- Field oriented data entry
- Stuff keyboard buffer
- 18 Mouse control functions
- Execute programs and batch files
- Disk error trapping
- Determine space available
- 40 functions to process characters and words
- Insert, delete, extract, index, translate
- Tested, easy-to-follow examples
- Demo programs with source code
- All source code included

Documentation:
Thorough, comprehensive, 260 pages

Compatible C Compilers:
Microsoft, Lattice, Computer Innovations, Aztec, Mark Williams, DeSmet, and Wizard

$185.00

Do Your Homework

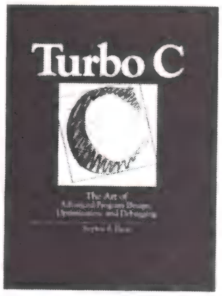
The library you buy can influence the rest of your programming life. We encourage you to do some checking before making a decision. When you've done your homework, you'll choose Essential. Call our support staff of experienced C programmers and find out before you buy how things will be after your check clears.

To order or for support call: 201-762-6965

For foreign orders contact:
England: Gray Matter Tel. (0364) 53499
Japan: Lifeboat Inc. of Japan Tel: 293 4711
West Germany: Omnitex Tel. 07623-61820

Essential Software, Inc.

P.O. Box 1003, Maplewood, New Jersey 07040



Turbo C: The Art of Advanced Program Design, Optimization and Debugging

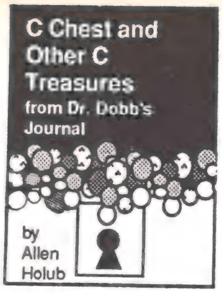
by Stephen R. Davis

Overflowing with example programs, this book fully describes the techniques necessary to skillfully program, optimize, and debug in Turbo C. Every topic and Turbo C feature discussed is fully demonstrated in Turbo C source code examples. Advanced topics such as pointers; direct screen I/O; inline statements in Turbo C; and how to intercept and redirect BIOS calls are all covered in depth. The author further demostrates these advanced topics by writing a RAM resident pop-up program in Turbo C. In addition, the author fully outlines the differences between Unix C and Turbo C; the transition from Turbo Pascal to Turbo C; and the superset of K&R C features implemented in Turbo C and included in the proposed ANSI C standard.

All programs are also available on disk with full source code. MS-DOS format.

► Book & Disk (MS-DOS) $39.95 Item #45-3

► Book $24.95 Item #38-0



C Chest and Other C Treasures from Dr. Dobb's Journal

Edited by Allen Holub

This comprehensive anthology contains the popular "C Chest" columns from *Dr. Dobb's Journal of Software Tools,* along with the lively philosophical and practical discussions they inspired, and other information-packed articles by C experts.

Topics covered include: pipes, wild-card expansion, and quoted arguments; sorting routines; command-line processing; queues and bit maps; utilities such as *ls, make,* and *more;* expression parsing; hyphenation; IBM cursor control and an Fget that edits; redirection; accessing IBM video display memory; trees; an AVL tree database package; directory traversal; sets; shrinking .EXE file images; hashing, expressions, and Roman numerals; and statistical applications of digital low-pass filters.

Other treasures include: a variable metric minimizer; Fgrep; a peephole optimizer; curve fitting with cubic splines.

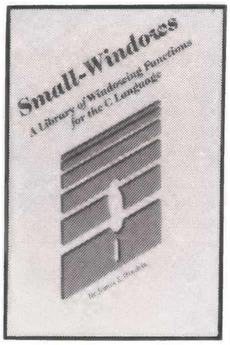
All subroutines and programs are written in C, and are available on disk with full source code. MS-DOS format.

► Book & Disk (MS-DOS) $39.95 Item #49-6

► Book $24.95 Item #40-2

Small-Windows: A Library of Windowing Functions for the C Language

by James E. Hendrix



Small-Windows is a complete windowing library for C (Microsoft 4.0, Small-C, and Lattice C.) The package includes:

- 18 video functions written in assembly language
- 7 menu functions that support both static and pop-up menus
- 41 window functions to clean, frame, move, hide, show, scroll, push, and pop windows.

A file directory illustrates the use of window menu functions and provides file selection, renaming, and deletion capability. Two test programs are provided as examples to show you how to use the library and the window, menu, and directory functions.

The **Small-Windows** package is available for MS-DOS systems, for the following compilers: Turbo C, Microsoft C Version 4.0, Small-C, and Lattice C. Documentation and full C source code is included.

► Manual & Disk (MS-DOS) $29.95 Item #35-6

SAVE 15%

Receive the **C Chest** book and disk, the **Turbo C** book and disk, and the **Small-Windows** manual and disk, all for only $93.95! You save 15%!

► C Chest/Turbo/Window Package
Item #168 $93.95

ORDER FORM

YES! Send Me:

Item #	Description	Price

For Small Windows, indicate:
☐ Microsoft C version 4.0 compiler
☐ Small C Compiler
☐ Lattice C Compiler
☐ Turbo C

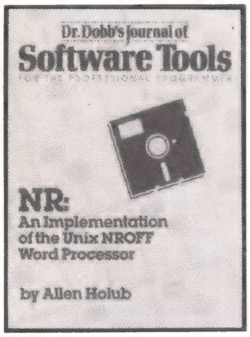
Subtotal ______
CA residents add sales tax ____%____
Add $2.25 per item for shipping ______
TOTAL ______

☐ Check enclosed. Make payable to M&T Publishing.
Charge my ☐ VISA ☐ M/C ☐ Amer. Exp.

Card # ______________________ Exp. Date ______
Name ______________________
Address ______________________
City ______________ State ______ Zip ______

3132A

Bring the Convenience of Unix-Like Features to Your MS-DOS Machine



NR: An Implementation of the Unix NROFF Word Processor

by Allen Holub

NR is a text formatter that is written in C and compatible with Unix's NROFF. Complete source code is included in the **NR** package so that it can be easily customized to fit your needs. **NR** also includes an implementation of the *-ms* (manuscript) macro package and an in-depth description of how *-ms* works. **NR** does hyphenation and simple proportional spacing. It supports automatic table of contents and index generation, automatic footnotes and endnotes, italics, boldface, overstriking, underlining, and left and right margin adjustment. **NR** also contains:

- extensive macro and string capability
- number registers in various formats, including Roman and Arabic numerals, both spelled out and in outline form
- diversions and diversion traps (macros that are triggered automatically)
- input and output line traps

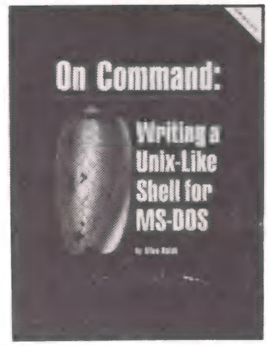
NR comes configured for any Diablo-compatible printer, as well as Hewlett Packard's ThinkJet, and LaserJet. It is easily configurable for most other printers. Both the ready-to-use program and full source code are included. For PC compatibles.

► Manual & Disk (MS-DOS)
$29.95 Item #33-X

TO ORDER: Return the Order Form on the reverse side

On Command: Writing a Unix-Like Shell for MS-DOS

by Allen Holub



This book and ready-to-use program demonstrate how to write a Unix-like shell for MS-DOS, with techniques applicable to most other programming languages as well. The book and disk include a detailed description and working version of the Shell, complete C source code, a thorough discussion of low-level MS-DOS interfacing, and significant examples of C programming at the system level.

Supported features include: read, aliases, history, redirection and pipes, Unix-like command syntax, MS-DOS-compatible prompt support, C-Shell-based shell scripts, and a Shell variable that expands to the contents of a file so a program can produce text that is used by Shell scripts. The Unix-like control flow includes: if/then/else; while; foreach; switch/case; break; continue.

The ready-to-use program and all C source code are included on disk. For IBM PC and direct compatibles.

► Book & Disk (MS-DOS)
$39.95 Item #29-1

/Util

by Allen Holub

When used with the Shell, this collection of utility programs and subroutines provides you with a fully functional subset of the Unix environment. Many of the utilities may also be used independently. You'll find executable versions of cat; cp; date; du; echo; grep; ls; mkdir; mv; p; pause; printevn; rm; rmdir; sub; and chmod.

The **Util** package includes complete source code on disk and all programs (and most of the utility subroutines) are fully documented in a Unix-style manual. For IBM PC and direct compatibles.

► Manual & Disk (MS-DOS)
$29.95 Item #12-7

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT 871 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

M&T Books
501 Galveston Dr.
Redwood City, CA 94063

SAVE 15%

Receive **On Command**, **Util** and **NR** together for only $85.95! You get all the convenience of Unix-like features, and save 15%!

► Unix-like Features Package
Item #167 $85.95

3132A

Listing One *(Listing continued, text begins on page 18.)*

```
XmitCount    EQU    SendingIMUR+1
XOFFTOcount  EQU    XmitCount+4
OVRcount     EQU    XOFFTOcount+4
RcvIntCount  EQU    OVRcount+4
XOFFcount    EQU    RcvIntCount+4
XONcount     EQU    XOFFcount+4
LongFrame    EQU    XONcount+4
ShortFrame   EQU    LongFrame+4
FrmCount     EQU    ShortFrame+4
NoHandCnt    EQU    FrmCount+4
CRCCount     EQU    NoHandCnt+4
LenErrCnt    EQU    CRCCount+4
BadDDP       EQU    LenErrCnt+4
PPCount      EQU    BadDDP+4
PPXoffCnt    EQU     PPCount+4
DeferXmit    EQU     PPXoffCnt+4
ABVarsEnd    EQU    DeferXmit+4
             ELSE
ABVarsEnd    EQU    SendingIMUR+1          ; end of AALAP variables
             ENDIF

***** file MPP.A *****

... section removed ...

;_____________________________________________________________________
;
; SCCConfig - set up the SCC for AppleBus
;_____________________________________________________________________

SCCConfig    LEA     OpenTbl,A0          ; A0 -> (common) open table
             CMP.B   #$FF,MacTypeByte    ; Mac or Lisa?
             BNE.S   @10                 ; Branch if Mac - configure it
             BSR     ToSCC               ; Configure SCC to major settings
             LEA     LOpenTbl,A0         ; A0 -> Lisa open table
@10          BRA     ToSCC               ; Configure SCC and return

... section removed ...

ToSCC        MOVE.L  SCCWr,A3            ; Point to SCC port B write registers
             IF      PortA THEN
             ADDQ    #ACtl,A3            ; Add in port A offset
             ENDIF
@10          MOVE    (A0)+,D0            ; Get next register number / control
word
             BEQ.S   CloseRTS            ; Zero is terminator
             MOVE.B  D0,(A3)             ; Put out register number
             ROR     #8,D0               ; Pickup control word
             MOVE.B  D0,(A3)             ; Set to SCC
             BRA.S   @10                 ; And keep going

... section removed ...

;_____________________
;
; Initialization tables
;_____________________

;
; SCC Initialization table - common between Mac and Lisa
; Entry format: .BYTE control-value, control-reg-number
; Taken from the Zilog SCC Application note, 00-2957-02

OpenTbl      DC.B    ResetOurPort,9      ; ($40 or $80) Reset port
             DC.B    $44,4               ; x16 clock, 1 stop, no parity
             DC.B    $0,2                ; Interrupt vector = $00
             DC.B    $C0,3               ; Rx is 8 bits, disable Rx
             DC.B    $E2,5               ; Tx is 8 bits, Disable Tx; DTR, RTS
on
             DC.B    $0,6                ; No address
             DC.B    $0,7                ; No Flag character
             DC.B    $0,10               ; NRZ
             DC.B    $56,11              ; Tx & Rx clock from BRG
             DC.B    $2,14               ; BRG source = PCLK, BRG off
; enables
             DC.B    $3,14               ; BRG on
             DC.B    $C1,3               ; Rx on
             DC.B    $EA,5               ; Tx on
; Interrupt controls
             DC.B    MouseInts,15        ; enable DCD ints (for mouse)
```

(continued on next page)

Introducing Network Version!

SEIDL VERSION MANAGER

Now SVM supports local area networks and tracks source revisions made by multiple users in both single-site and multi-site configurations.

Plus...

- Archive Database Tracks Source (and Binary) File Revisions
- Audit Trail Reporting Provides Info on Project's Development
- Revision Branches Allow Multiple Courses of Development
- Revision Merging and Deleting Provide Flexibility in Archive Maintenance
- User IDs, Privilege Settings & Passwords Help Resolve Access Conflicts and Maintain Project Integrity
- Optional Text Compression Reduces Storage Requirements
- Menu Driven Shell Makes SVM Easy to Use
- Single-Site: $299.95*
- 5-site LAN: $1000 (extendible)

Now Builds Dependencies!

SEIDL MAKE UTILITY

New program, called SMKgen, automatically constructs a dependency file by analyzing the files in a project.

Plus...

- Structured Language Used to Define Dependencies
- Rich Command Set with Over 20 Different Statements
- Ability to Handle Nested Include Files and Library Dependencies
- Performance & Functionality not Found in UNIX Make or Clones
- SMK Only: $99.95*
- SMKgen: Add $50.00

CALL TODAY

1-313-662-8086

Visa/MC/COD Accepted
Dealer Inquiries Invited
*Plus postage and Handling

SEIDL COMPUTER ENGINEERING
3106 Hilltop Dr., Ann Arbor, MI 48103

CIRCLE 114 ON READER SERVICE CARD

Listing One *(Listing continued, text begins on page 18.)*

```
            DC.B    $10,0               ; reset external ints
            DC.B    $10,0               ; reset external ints (twice)
            DC.B    $13,1               ; Tx, Rx, Ext int enable
            DC.B    MIE,9               ; Master Interrupt Enable
            DC.W    0                   ; *** End of table ***

            IF      RAM THEN            ; Only need Lisa table if RAM-based
;
; SCC initialization table for Lisa
; Port A uses PCLK (@4.0 MHz TTL) to drive BRG; Port B uses 3.6864 MHz Xtal
;
            IF      PortA THEN          ; configuration for Port A
LOpenTbl    DC.B    $00,14              ; turn off BRG
            DC.B    $6A,5               ; enable TX, RTS; DTR low
            DC.B    $56,11              ; TTL clock, tx and rx use BRG
            DC.B    $02,14              ; Use PCLK to feed BRG, BRG off
            DC.B    $03,14              ; BRG on
            DC.W    0
            ELSE                        ; configuration for PortB
LOpenTbl    DC.B    $00,14              ; turn off BRG
            DC.B    $6A,5               ; enable TX, RTS; DTR low
            DC.B    $D6,11              ; Crystal clock, tx and rx use BRG
            DC.B    $00,14              ; Use crystal to feed BRG, BRG off
            DC.B    $01,14              ; BRG on
            DC.W    0
            ENDIF

;
; Mac initialization data (first is the post-transmitting SCC string)
;
MacInitData DC.B    5,MDisTxRTS         ; ($60) Turn off drivers
            DC.B    14,ResetClks        ; ($41) Reset missing clocks flag
            DC.B    3,EnbRxSlv          ; ($DD) Enable receiver
            DC.W    $2100               ; SR to enable SCC interrupts
            DC.B    AbortDelay,0        ; Delay to send out abort bits (3.2B)
            IF      RAM THEN
;
; Lisa initialization data
;
LisaInitData DC.B   5,LDisTxRTS         ; ($E2) Turn off drivers
            DC.B    14,ResetClks        ; ($41) Reset missing clocks flag
            DC.B    3,EnbRxSlv          ; ($DD) Enable receiver
            DC.W    $2500               ; SR to enable SCC interrupts
            DC.B    34,0                ; Just delay this much on Lisa (3.2B)
            ENDIF

***** file LAP.A *****

;__________________________________________________________________________
;
; LAP.TEXT - the LAP part of AALAP
;
; April-August, 1984
; Alan Oppenheimer and Larry Kenyon
;
; Rich Brown, Dartmouth College
; May 1987
;
; Version 1.2a6 Created qWDSptr to point at queued WDS  21 May 87   reb
; Version 1.2a5 Always check that TBMT is true before sending  19 May 87   reb
; Version 1.2a4 TintHnd, VBLHnd, RintHnd now call TxNextCh; only TintHnd
; clears interrupts (as it should be)   14 May 87   reb
; Version 1.2a3 Prefetching warning dialogs doesn't work; backed out  10 May 87 reb
; Version 1.2a2 tWDSptr now determines whether we're sending a frame;
; DoWarn now doesn't read the resource file  8 May 87   reb
; Version 1.2a1 Removed queueing from LAPWrite.  LAPWrite no longer
; allocates memory, so it won't fail if called from
; interrupt handling.   19 Apr 87   reb
; Version 1.1b2 Changed noAnswer to -95 (so it can be handled like excessCollsns)
; LAPWrite returns noAnswer if AALAP not up; (30 Mar 87)
; GetNNNN  returns noAnswer or PortNotCF;
; Changed LAPWrite to return ddpLenErr if too long
; Version 1.1b1 Fixed PollProc to be more agressive about sucking chars
; from the SCC; added -1 SendChar value (sends Break);
; fix Initcursor bug in Dowarn 16 & 30 Dec 86   reb
; Version 1.1a1 Output an Xoff if called by PollProc during an input message
; 3 Nov 86   reb
; Version 1.0b2  Changed to set up SCC properly for Lisa  15 Oct 86
; (still has intermittent hangups, tho -- not diagnosed)
; Version 1.0b1 Changed last_valid_frame timer to 30 seconds; always send
; UR, even after un-matchable IM address
```

```
; Version 1.0a3 Fixed Status return buffer bug; SetBaud now takes actual
;                    baud rate; added GetLAPStatus call; copy entire message
; Version 1.0A2 Added alerts for NoAnswer, PortNotCf    (17 Jul 86    reb)
; Version 4.2   Int handlers now do IUS etc. more carefully (4 Jul 86)
; Version 4.1   Now escapes either parity Xon and Xoff (21 Apr 86)
; Version 4.0   First cut at AALAP (26 Oct-14 Dec 85)
... section removed ...
;
; COPYRIGHT (C) 1984 APPLE COMPUTER
;_______________________________________________________________________

... section removed ...

;_______________________________________________________________________
;
; MReInit - Control call to reinit AALAP and the SCC
;_______________________________________________________________________

MReInit     bsr.s AALAPWarm            ; Warm start ourselves
            BRA   AbusExit             ; and return
;_______________________________________________________________________
;
; AALAPCold -- cold start for AALAP; called only once
;_______________________________________________________________________

AALAPCold
;
; Allocate the input buffer (This should be alloc above BufPtr, not sysheap)
;
            move.l  #maxLAPFrmLen,D0     ; get an AALAP input buffer
            _newptr ,SYS                 ; from the system heap
            bne.s  WarmRTS               ; exit if bad
            move.l  A0,LAPInBuf(a2)      ; otherwise save its pointer
;
; Clear out LAP variables
;
            clr.l   WDSPtr(a2)
            clr.l   tWDSptr(A2)
            clr.l   LAPWrtRtn(A2)
            clr.w   SysNetNum(a2)
            clr.b   SysLAPAddr(a2)
            clr.l   SavePS(A2)
            sf      AALAPup(a2)
            sf      AALAPstuck(a2)
;
; Setup SCC for AALAP
;
            BSR     SCCConfig            ; Configure the SCC for Async
AppleTalk
            move    #9600,D0             ; and set up for 9600 baud
            bsr      Set_Baud            ;
;
; Reset all the LAP variables which don't irrevocably change the
; state of the driver.  This routine can be called any time, only
; killing the current message(s) in progress.
;
AALAPWarm   move.l  Ticks,D0
            move.l  D0,LastXmit(a2)
            move.l  D0,LastRcv(a2)
            lea     BusyBuf(a2),A0
            move.l  A0,BusyStash(a2)
            move.l  A0,BusyFetch(a2)
            move.w  #$FFFF,LastRxCh(a2)
            clr.l   tWDSptr(A2)
            clr.l   qWDSptr(A2)
            clr.w   TxCount(a2)
            sf      RcvdXoff(a2)
            sf      InpState(a2)
            sf      EscIn(a2)
            sf      SendingIMUR(A2)
            sf      stillBusy(a2)
            sf      nFrmChr(a2)
            sf      nCRC(a2)
            sf      nXon(A2)
WarmRTS     rts

            EJECT
;_______________________________________________________________________
;
; Status - handle driver status request
;_______________________________________________________________________
```

(continued on next page)

DAN BRICKLIN'S DEMO PROGRAM ONLY $74.95

Read what they're saying about this popular program for prototyping and demo-making:

"A winner right out of the starting gate. After you use DEMO once, you'll wonder how you got along without it."
—PC Magazine

"Everybody who writes software, either commercially or for in-house applications, should immediately order a copy. Period. No exceptions."
—Soft·letter

Product of the Month
—PC Tech Journal

Thousands of developers and most of the largest and best known software companies are using this program. You can, too. Act now!

NEW TUTORIAL! JUST $49.95

The perfect companion to the Demo Program. The Tutorial helps you learn the ins and outs of its basic and advanced features. Complete with a 96 page manual containing step-by-step instructions, diskette, and function key template.

ORDER NOW!
1-800-CALL-800 x8088

VISA MasterCard AMERICAN EXPRESS

Use 800-number for orders only.
Questions, special shipping, etc., call **617-332-2240.**
No Purchase Orders. Massachusetts residents add 5% sales tax. Outside of the U.S.A., add $15.00.
Requires 256K IBM PC/Compatible, DOS 2.0 or later. Supports Monochrome, Color Graphics, and EGA Adapters (text mode only). The Tutorial requires the Demo Program.

SOFTWARE GARDEN, INC.

Dept. D
P.O. Box 373, Newton Highlands, MA 02161
CIRCLE 314 ON READER SERVICE CARD

Listing One *(Listing continued, text begins on page 18.)*

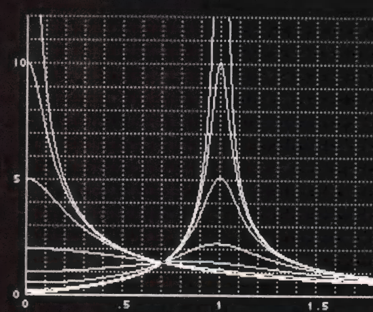
```
            _SUBR                             ; no one better call this...
Status      MOVE.L  MPPVars,A2                ; A2 -> our variables
            MOVEQ   #StatusErr,D0             ; Assume a status error
            lea     CSParam(A0),A1            ; point at the CSParam buffer
            move.w  CSCode(A0),D1             ; and get the CScode

            IF      Stats THEN
            CMP.W   #GetStats,D1              ; Clear stats command?
            BNE.S   @1                        ; check for "What's my Name?" if not
            move.w  CSParam(A0),A1            ; CSParam contains a pointer to buffer
            MOVE    SR,-(SP)
            MOVE    #SCCLockout,SR            ; exclude interrupts to keep stats
clean
            ADD     #StatsStart,A2            ; point to stats we keep
            MOVEQ   #(StatsLgCnt-1),D0
            MOVEQ   #0,D1                     ; zero for faster clearing
@0          MOVE.L  (A2),(A1)+                ; return current value
            MOVE.L  D1,(A2)+                  ; then zero count
            DBRA    D0,@0
            MOVE    (SP)+,SR
            bra.s   AbusExit
            ENDIF

@1          cmp.w   #GetMyName,D1             ; is this a "what's my name" call?
            bne.s   @2                        ; go if not
            Move.l  MPP+18,(A1)+              ; move Pascal string from front of driver
            move.b  MPP+22,(A1)               ; to beginning the buffer (5 chars)
            bra.s   @4                        ; and exit with good status

@2          cmp.w   #GetChar,D1               ; is this a "get last char" call?
            bne.s   @3                        ; go if not
            move.w  LastRxCh(a2),(A1)         ; copy the character (word)
            move.w  #$FFFF,LastRxCh(a2)       ; and flag the character
            bra.s   @4

@3          cmp.w   #GetLAPStatus,D1          ; is this a "get LAP status" call?
            bne.s   AbusExit                  ; go if not
            move.b  AALAPup(A2),(A1)+         ; AALAPup?
            move.b  AALAPstuck(A2),(A1)+;AALAPstuck?
            move.w  AALAPbaud(A2),(A1)        ; What's the baud rate?
@4          clr.l   D0                        ; return good status

AbusExit    MOVE.L  MPPDCE,A1                 ; Make sure A1 has DCE address
AbusExA1    MOVE.L  JIODone,-(SP)             ; This is how we exit (Prime, Control,
Status)
AbusRTS     RTS
            SUBEND  'MYSTATUS'                ; this marks the AbusExit

Prime       BCLR    #DrvrActive,DCtlFlags+1(A1) ; *** V2.0C Fix Mac ROM bug
***
            RTS                                       ; *** V2.0C Fix Mac ROM bug
***
            EJECT
;____________________________________________________________________
;
; MGetNNNN -- Do the NNNN, using the current values of SysLAPAddr and
;             SysNetNum.  Return bad status if it didn't work.
;
; On entry: A2 -> BPP variables
; On exit:  D0 = noErr (0) if we succeeded,
;                PortNotCF (-98) or
;                noAnswer  (-95) if not
;____________________________________________________________________

MGetNNNN    bsr.s   Get_NNNN                  ; Use them just as they are
            bra.s   AbusExit                  ; return from the control calls

tries       EQU     -2                        ; counter for the tries
endtime     EQU     -6                        ; end time

Get_NNNN    _SUBR   6
            move.w  Ticks+2,RandomSeed(a2) ; randomize things
            move.b  SysLAPAddr(a2),D0         ; Node number in D0
            move    SysNetNum(a2),D1          ; Net number in D1
            sf      AALAPup(a2)               ; we're not up yet
            sf      AALAPstuck(a2)            ; and we're not in trouble either
            move    #4,tries(a6)              ; tries counter (4 tries)
@10         move.l  Ticks,D2
            add.l   #30,D2                    ; set endtime to the current time+30
            move.l  D2,endtime(a6)            ; remember the ending time
            moveq   #qlapIM,D2                ; get the lap type
```

```
           move.w   SysNetNum(a2),D1   ; get the Net number
           move.b   SysLAPAddr(a2),D0  ; and the node number
           bsr      SendIMUR           ; and send it
@20        clr      D0                 ; good status if things are OK
           tst.b    AALAPup(a2)        ; did the magic work?
           bne.s    NNNNexit           ; go if so
           tst.b    AALAPstuck(a2)     ; is there an irreconcilable difference?
           bne.s    NNNNstuck
           move.l   endtime(a6),D2
           cmp.l    Ticks,D2           ; otherwise, check the timer
           bpl.s    @20                ; loop if not timed out
           sub      #1,tries(a6)       ; decr the counter
           bgt.s    @10                ; loop if non-zero
           moveq    #noAnswer,D0       ; They don't want to talk
           bra.s    NNNNexit

NNNNstuck  moveq    #PortNotCF,D0      ; They talk but say bad things

NNNNexit   tst.w    D0                 ; set CC
           _Subend  'GETNNNN '         ; and return
           EJECT
;_____________________________________________________________________________
;
; MPutChar -- Kill output and send the char pointed to in the control call
;
; Entry:   A0 -> IOQelement
; Exit:    Return status is 0000  if noErr,
;                          BadIO  if timed out waiting for TBMT
;_____________________________________________________________________________

SendBrk    equu     $12                ; Sends Break (w/RTS) when sent to WR5

MPutChar   bsr.s    Put_Char
           bra      AbusExit           ; and exit

Put_Char    _SUBR
           move.w   CSParam(a0),D0     ; get the character (in an integer)
           bmi.s    @10                ; if it's 0..255,
           bsr      SendChar           ;    output the character
           bra.s    @20                ; and quit

@10        lea      BreakTbl,A0        ;
           move.b   #SendBrk,(A0)      ; set the break bit in WR5
           bsr      ToScc              ;
           move.l   #10,A0             ; wait 10 ticks
           _delay
           move     #$EA,D0            ; Enable Tx, DTR, RTS
           CMPI.B   #$FF,MacTypeByte   ; Mac or Lisa?
           BNE.S    @15                ; Branch if Mac (PortA & PortB are same)
           move     #$6A,D0                     ; Lisa doesn't assert DTR
           @15lea   BreakTbl,A0
           move.b   D0,(A0)              ; and turn the Break off
           bsr      ToSCC
           clr      D0

@20        SUBEND 'MPUTCHAR'

breaktbl   dc.b     0,5      ; THIS WON'T MAKE ROMMABLE CODE
           dc.w     0000
;____________________________________________________________________
;
;  MSetBaud -- send the (integer) value in the CSParamblk to SCC as its Baud
Rate
;  Entry:  A0 -> IOQelement
;  Exit:   noErr if aok
;          -1    if requesting 19,200 baud on a Lisa, port A (cannot be done)
;____________________________________________________________________
; THIS WON'T MAKE ROMMABLE CODE!
BaudConsts  DC.B    2,14               ; turn off BRG (so it doesn't count for
a while)
lsBaudVal   DC.B    0,12               ; LSByte of BRG
msBaudVal   DC.B    0,13               ; MSByte of BRG
BaudSrc     DC.B    0,14               ; turn it on again, with proper baud
source
            DC.W    0000               ; end of constant string

BaudTable   DC.W    1200,94,102        ; 1200 baud, Mac&LisaB , LisaA BRG
constants
            DC.W    2400,46,50         ; 2400 baud
            DC.W    4800,22,24         ; 4800 baud
            DC.W    9600,10,11         ; 9600 baud
```

(continued on next page)

isys forth

FOR THE APPLE IIgs™
AND OTHER APPLE® II MODELS



Parallel Resonance with Damping.
BASIC 213 sec. ISYS FORTH 20 sec.
IIgs ISYS FORTH 5 sec.

ISYS FORTH-83 is designed especially for scientific and engineering applications. Each system includes versions for the IIgs and for older Apple IIs. A version for older Apples equipped with a 65802 or a 65816 is also included.

FASTEST. Uses subroutine threading with optional use of macros.

TURTLE AND CARTESIAN GRAPHICS, including character sets and double hires. The IIgs version supports the new super hires graphics.

FLOATING POINT WITH TRANSCENDENTALS, single and double precision. And double precision integer math, including D*/.

MACRO ASSEMBLER for the 6502, 65C02, 65802 and 65816.

MODEST MEMORY REQUIREMENTS. No IIgs memory expansion is needed.

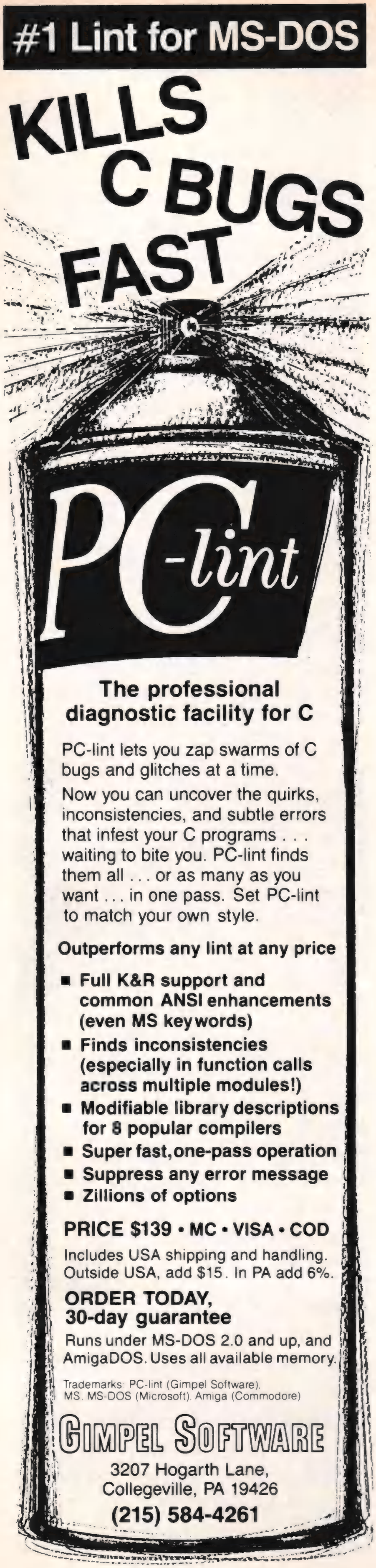
3.5 AND 5.25 DISKS are supported.

PRICE: $99. This includes all three versions of ISYS FORTH, a 150-page manual and all of the above features. There is no charge for shipping.

ILLYES SYSTEMS PO Box 2516, Sta A
Champaign, IL 61820 Phone: 217/359-6039

Apple and Apple IIgs are trademarks of Apple Computer.

CIRCLE 372 ON READER SERVICE CARD



Listing One *(Listing continued, text begins on page 18.)*

```
            DC.W    19200,4,-1          ; 19200 baud (but not for Lisa Port
A...)
BaudTblEnd  DC.W    -1                  ; sentinel
;
MSetBaud    move    CSParam(a0),D0      ; get the (integer) baud rate
            bsr.s   Set_Baud
            bra     AbusExit

Set_Baud    _SUBR                       ; D0 contains the actual desired baud
rate
            move.w  D0,AALAPbaud(A2)    ; save the current baud rate
            lea     BaudTable,A0        ; point at the table
@10         cmp.w   (a0),D0             ; does it match?
            beq.s   @30                 ; go if so
            addq.l  #6,A0
            tst.w   (A0)                ; are we done?
            bpl.s   @10                 ; loop if we didn't hit the sentinel
            moveq   #-1,D0              ;
            bra.s   @50                 ; and bail out

@30         moveq   #3,D1               ; set up for Mac port A/B (PCLK/BRG on)
            CMPI.B  #$FF,MacTypeByte    ; Mac or Lisa?
            BNE.S   @40                 ; Branch if Mac (PortA & PortB are
same)
        IF  PortA THEN                  ; Lisa ports A/B differ; Macs don't
            addq.l  #2,A0               ; bump to Lisa PortA column
        ELSE
            moveq   #1,D1               ; Lisa portB works from Xtal, not PCLK
        ENDIF
@40         move.w  2(a0),D0            ; get value BRG (-1 if 19,200 on Lisa)
            bmi.s   @50                 ; exit if negative

; D0 now contains the value for the BRG
            lea     BaudConsts,a0       ; point at the constants
            move.b  d0,lsBaudVal-BaudConsts(a0) ; save the LSByte of the BRG
            ror     #8,d0
            move.b  d0,msBaudVal-BaudConsts(a0) ; and the MSByte of the BRG
            move.b  d1,BaudSrc-BaudConsts(a0)   ; and the source for BRG
            bsr     toSCC               ; and output it
            clr     d0
@50         _SUBEND 'MSETBAUD'
            EJECT
;_______________________________________________________
;
; MWriteLAP - write out a LAP packet
;
;   Call:
;       A0 -> IO queue element
;       A1 -> WDS. First entry must start as follows:
;                 +------------------+
;                 | Destination addr|
;                 +------------------+
;                 |                  |     [ for source addr ]
;                 +------------------+
;                 |  LAP type code   |
;                 +------------------+
;                 :                  :
;       A2 -> local variables
;
;   Return:
;       D0 = error code
;
; NOTE: for MPP, first two data bytes must be length
;_______________________________________________________

MWriteLAP   MOVE.L  2(A1),A0            ; A0 -> first WDS entry
            MOVEQ   #LAPProtErr,D0      ; Assume an error (2.3F)
            TST.B   LAPType(A0)         ; Make sure protocol is a valid one
            ble.s   MWRLAPex            ; Return error if not
            MOVE.B  LAPDstAdr(A0),D2    ; D2 = destination address
            bsr.s   LAPWrite            ; Write out the packet
MWRLAPex    bra     AbusExit
            EJECT
;_______________________________________________________________________
;
; LAPWrite - send a packet out an Async port.  Called both by MWriteLAP
; and DDPWrite.
;
;   Call:
;       A1 -> WDS (first entry must start as in MWriteLAP above)
;       A2 -> local variables
```

```
;        D2 = LAP destination address
;
;   Return:
;        D0 = noErr or the error code
;        Uses D1-D3,A0,A1,A3
;
; Save the WDS passed in
; If AALAP isn't up, return noAnswer
; Next, check the length of the frame for <= 603 bytes; return error if bad
; If we're currently sending a frame:
; if it's an IM/UR, simply return (WDS will be sent when done)
; if it's not, then stop (somehow we got two frames to send from DevMgr)
; If interrupts are on
;    Update PollProc pointer if it needs it
;    Check that the AALAP is still working, sending IM/UR if necessary.
; Start sending the frame
;
; This code relies on the Device Manager for queuing.  Here's how it works:
;
; General Rule #1: All operations initiated by the device manager
; ultimately return to the DevMgr through jIOdone.
;
; General Rule #2:  All async operations which cannot complete immediately
; return thru a RTS.  When the operation does complete, the (interrupt)
; routine can go thru jIOdone.
;
; Specific AppleTalk Rule #1:  All callers of LAPWrite have bra AbusExit
; code right after the call to LAPWrite.  This eventually jumps to jIOdone.
;
; Specific AppleTalk Rule #2:  Since they've taken care of the details,
; LAPWrite only has to remember two things: If we finish, we can return
; to our original caller (by jumping thru LAPWrtRtn to go to the device
; manager);  If we don't finish, we should return to the caller's caller
; (which called the device manager in the first place).  Whew!
;_____________________________________________________________________________

LAPWrite    move.l    (SP)+,LAPWrtRtn(A2) ; save the caller's adrs
            move.l    A1,WDSptr(A2)       ; and the frame we're asked to send

            move.w    #noAnswer,D0
            tst.b     AALAPup(a2)         ; is the AALAP up?
            beq       LAPWexit            ; exit if bad
;
; Next compute the length of the WDS -- exit if it's bad
;
            move.l  A1,A0                  ; get the WDS pointer

            clr.l   D2                    ; D2 = number of data bytes in frame
            clr.l   D1                    ; D1 = number of segments in WDS
            cmp.w   #2,(a0)               ; is first segment too short?
            ble.s   LAPWexit              ; go if it is
@20         tst.w    (a0)                 ; is WDS length = 0?
            beq.s   @30                   ; go if so
            add.w   (a0),d2               ; add in this length
            addq    #1,d1                 ; incr the segment counter
            addq.l  #6,A0                 ; bump the WDS pointer
            bra.s   @20

; D2 is the length of the message we've been asked to send
; D1 is the number of segments we've been presented with
; (A1 still has WDSptr)

@30         moveq   #LAPProtErr,D0
            tst.l   d1                    ; is D1 (number of segments) < 1?
            ble.s   LAPWexit              ; go if so (error)
            moveq   #ddpLenErr,D0
            cmp.w   #603,D2               ; is the length > 603 (3 LAP + 600
data)
            bgt.s   LAPWexit              ; go if it's bad
;
; we can try to send WDS in A1 -- are we currently sending a frame?
;
            tst.l   tWDSptr(a2)           ; are we presently sending a frame?
            beq.s   @40                   ; go if not
            tst.b   SendingIMUR(A2)       ; is it an IM or UR?
            beq.s   @35                   ; go if not
            tst.l   qWDSptr(A2)           ; is one already queued?
            bne.s   @35                   ; go if so (stop)
            move.l  A1,qWDSptr(A2)        ; save the (queued) WDS pointer
           _statcount DeferXmit
            rts
```

(continued on next page)

ZyINDEX Your personal researcher™

FULL TEXT RETRIEVAL

FIND

Searches 5,000 text files in 5 seconds.

The ultimate personal computer information retrieval software.
Find any information created by the popular word processors or ASCII files, in seconds-without having to set up a database or do programming. Scan a 20 Mbyte hard disk or a collection of floppies and instantly display all occurrences of a name, a product, a phrase, a number, or anything else you need to find. Save hours of manual searching.

TEXT

FEATURES
• **Comprehensive searches** create a search request with any combination of AND, OR, NOT, and WITHIN (WITHIN refers to how far apart two words or phrases can be — up to 30,000 words).
• **Wild Cards,** for example: type "micro*" to get (microcomputer, microcomputing, micro-processor, etc.). • **Mark and Save** retrieved information to create a new file.
• **Highlights** search-words and phrases in retrieved text. • **On Line Help Menus.**
• **Find Function** automatically displays search topic in the retrieved file. • **Phrase Search,** in addition to single word search.

SYSTEM REQUIREMENTS MINIMUM
• 384K • Two Disk Drives • DOS 2.0 or above • Designed for IBM PC, XT, AT, compatibles, and most MS-DOS computers

FAST

ZyINDEX Personal $95
Searches up to 325 files
ZyINDEX Standard $145
Searches up to 500 files
ZyINDEX Professional $295
Searches up to 5,000 files
ZyINDEX Plus $695
Searches up to 15,000 files with LAN capabilities.
30 Day Money Back Guarantee

ZyLAB™ Corporation
233 East Erie Street
Chicago, Illinois 60611 (312) 642-2201
(800) 544-6339 For orders and information

CIRCLE 329 ON READER SERVICE CARD

Listing One *(Listing continued, text begins on page 18.)*

```
@35         pea      AALAP2in1            ; point at the string
            DC.W     $ABFF                ; and trap 'em (in lieu of $A9FF)

;
; WDS in A1 is OK to send now: if interrupts enabled,
;    update PollProc and check time since last good frame
;
@40         move     SR,D0
            and      #$70,D0              ; is the interrupt mask <> 0?
            bne.s    SendWDSptr           ; just send it
;
; Update our local PollProc pointer
;
            move     SR,-(A7)             ; save the state
            move     #SCCLockout,SR       ; turn off interrupts
            lea      myPollProc,A1        ; A1 -> our PollProc
            move.l   PollProc,D0          ; get the current PollProc address
            cmp.l    D0,A1                ; have we already updated it?
            beq.s    @50                  ; go if we have
            move.l   D0,SavePS(A2)        ; else update our saved copy
            move.l   A1,PollProc          ; and point the real PollProc at us
@50         move     (A7)+,SR             ; and re-enable
;
; check for (Ticks - LastRcv) > 1800 - see if they're still there
;
            move.l   Ticks,D0             ; have we received a frame recently?
            sub.l    LastRcv(a2),D0
            cmp.l    #1800,D0             ; (ticks - LastRcv) > 1800 (30 sec)?
            bmi.s    SendWDSptr           ; go if not (send it)
            bsr      Get_NNNN             ; do the IM/UR stuff
            beq.s    SendWDSptr           ; go if it worked
            move.w   D0,-(SP)             ; otherwise, save the status
            bsr      DoWarn               ; else, warn them
            move.w   (SP)+,D0             ; and return bad status
;
; Come here if we need to return immediately (status is in D0)
;
LAPWexit    move.l    LAPWrtRtn(A2),A0    ; this'll get 'em to IOdone
            jmp       (A0)                ; sooner or later

AALAP2in1        dc.b 24
            dc.b     'AALAP - TWO MSGS AT ONCE'
            align 2
            EJECT
;__________________________________________________________________
;
; SendFrame -- Starts off transmission of a frame
;
; A0 points to the WDS of the frame to send
;
; SendFrame sets all the pointers, etc.  and then sends the FrameChar
; ($A5).  The Transmit Interrupt Handler ships all the remaining bytes
; as they are needed.
;
;__________________________________________________________________

SendWDSptr  move.l   WDSptr(A2),A0        ; get the WDS to send
SendFrame   move.w   (a0)+,D0             ; D0 = the length of the 1st segment
            move.l   (a0)+,a1             ; a1 -> the first byte of 1st segment
            move.l   a0,tWDSPtr(a2)       ; and save the pointer to rest of WDS
            subq     #2,D0                ; Finagle the length and address
            move     D0,TxCount(a2)       ;   of the segment (AALAP doesn't
            addq.l   #2,A1                ;   send dest and source node)
            move.l   a1,LAPFetch(a2)      ;
            st       nCRC(a2)             ; we'll need to send a CRC
            st       nFrmChr(a2)          ;    and a closing FrmeChar
            sf       EscOut(a2)           ; clear the Escape flag
            clr      OutputCRC(a2)        ; and the CRC
            moveq    #qFrmChar,D0         ; load a FrameChar
            bra.s    SendSCC              ; and kick off the frame
           EJECT
;__________________________________________________________________
;
; LAPSend -- send the next byte in the LAP frame
;
; This routine checks to see if we're flow-controlled, if not, it
; gets the next char, accumulates the CRC, generates DLE's as
; required, and calls the routine to place the byte in the SCC.
;
; It works from LAPFetch(a2), and advances it (and decrements TxCount)
; as necessary.
```

(continued on page 74)

Step into the Future

StonyBrook

Introducing

MODULA-2

for 8086 family machines

SPEED

- Compiles 5K lines / minute on PC/AT
- Runs Sieve faster than Microsoft C V4.0

FLEXIBILITY

- Generates Microsoft standard objects
- Supports 6 memory models + mixed model
- Interfaces directly to other DOS languages (you don't have to throw out your C code!)

POWER

- Full Modula-2 language
  + Array and record constants
  + Substrings and sub-arrays
  + Tailorable procedure calling

REPERTOIRE by PMI, indexed file and screen management package included free!

PLUS OS/2 and Microsoft Windows compatible **NOW**

The generated code is compatible with OS/2 and Microsoft Windows. Runtime libraries will be available soon.

StonyBrook INC SOFTWARE

Forest Road
Wilton, New Hampshire 03086
(603) 654-2525

Compiler and DOS runtime library objects only: **$195**

Above plus editor, source debug, make utility, and runtime library sources: **$345**

No other Modula-2 compiler comes close!

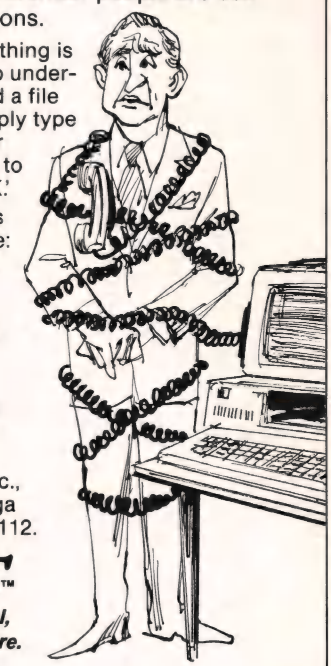
Add $5 shipping and handling in North America, $15 for overseas orders. VISA and MasterCard accepted.

Special introductory offer: Purchase and pay for the complete system before Oct. 31, 1987 and get The Watcher execution profiler by Stony Brook Software — a $60 value — free.

CIRCLE 246 ON READER SERVICE CARD

PC Communications Tying You Up in Knots?

Complicated screens, multi-level menus, windows on top of windows. It's no wonder people are confused about communications.



Not so with Move-It. Everything is straight forward and easy to understand. Say you want to send a file to another computer — simply type 'SEND.' Want to call another system? Say 'CALL.' To talk to your mainframe, type 'TALK.'

Of course, Move-It supports all the functions you require: multiple protocols, macros, wildcards, etc., as well as advanced functions, like automatic file compression, scripting, on-screen help, and unattended operation.

So if you're tired of fighting your communications program, give us a call.

Woolf Software Systems, Inc., 22048 Sherman Way, Canoga Park, CA 91303, (818) 703-8112.

MOVE-IT™

Move-It Program and Manual, $150.00. Call for free brochure.

CIRCLE 240 ON READER SERVICE CARD

GRAPHICS TOOLS FOR C and FORTRAN-77 PROGRAMMERS

GRAFLIB — Source Included — **$175.00**
FORTRAN-77 & C callable screen graphics routines. Supports CGA, EGA and Hercules Graphics Cards.

PLOTHP — Source Included — **$175.00**
Plotting routines for Hewlett-Packard & HPGL compatible plotters. Available in FORTRAN-77 & C.

PLOTHI — Source Included — **$175.00**
Plotting routines for Houston instruments DM/PL compatible plotters. Availale in FORTRAN-77 & C.

FORTLIB — Source Included — **$125.00**
Add C-Like power to your FORTRAN-77 Programs with our 120+ FORTRAN callable assembly and FORTRAN-77 Routines.

NO ROYALTIES

Call or write for a Brochure

SUTRASOFT
P.O. Box 1733
Sugar Land, TX 77487-1733
(713) 491-2088

CIRCLE 395 ON READER SERVICE CARD

Listing One *(Listing continued, text begins on page 18.)*

```
;
; If we sent a char, then we set SentChar(A2) to true
;_______________________________________________________________

LAPSend      tst.b    RcvdXoff(a2)          ; are we flow controlled?
             bne.s    LAPSendRTS            ; go if so

             move     TxCount(a2),D3        ; get the remaining length
             ble.s    LAPBadCount           ; go if zero or negative
             cmp.w    #maxLAPFrmLen,D3      ; check its length
             bgt.s    LAPBadCount           ; go if too big

             move.l   Ticks,LastXmit(a2)    ; remember when we last sent a char
             subq     #1,D3                 ; decr the count
             move.l   LAPFetch(a2),a0       ;
             move.b   (a0)+,D0              ; and fetch the character, bumping the ptr

             tst.b    EscOut(a2)            ; are we escaping this char?
             bne.s    @15                   ; go if yes -- it's already in CRC
             lea      OutputCRC(a2),a3      ; point at the output CRC Accumulator
             bsr      NextCRC               ; accumulate the un-processed char

             cmp.b    #DLE,D0               ; test for DLE, Xon, Xoff, FrameChar
             beq.s    @10                   ; go if it's a special one
             cmp.b    #FrameChar,D0
             beq.s    @10
             move.b   D0,D1
             and.b    #$7F,D1               ; is it a XON or XOFF (either parity)?
             cmp.b    #Xoff,D1
             beq.s    @10
             cmp.b    #Xon,D1
             bne.s    @20                   ; go if it's just a normal character

@10          st       EscOut(a2)            ; remember that we're escaping
             moveq    #DLE,D0               ; data to send is a DLE
             bra.s    SendSCC               ;  (and don't update the pointer/len)

@15          eor      #$40,D0               ; come here if we're escaping this char
@20          move.l   a0,LAPFetch(a2)       ; update the pointer
             move     D3,TxCount(a2)        ;    and the remaining length
             sf       EscOut(a2)            ;
;                                           ; D0 has the next char to send
;            bra.s    SendSCC               ; and send the character
;
; SendSCC -- sends D0 to the SCC Write Data Register
;            Assumes that SCC is ready (TBMT is true)
; Returns D0 = 0
; uses A1
;
SendSCC      st       SentChar(A2)          ; remember we sent a char
             move.l   SCCWr,a1              ; point at the SCC Write Control
             IF       PortA THEN
             addq.l   #ACtl,a1              ; add in the offset for Port A
             ENDIF
             move.b   D0,SCCData(a1)        ; output the character
             moveq    #0,D0                 ; clear the return status
LAPSendRTS   rts                            ; and return
LAPBadCount  pea      BadCntStr
             DC.W     $ABFF                 ; Trap 'em (not $A9FF)
             rts
BadCntStr          DC.B       10
             DC.B     'Bad length'
             align 2
             EJECT
;_______________________________________________________________
;
; SendChar -- Synchronously wait for TBMT and send another character
;       Use Ticks to watch for 1/2 sec timeout, so we don't hang forever
;
; Entry:          D0 = char to send
; Exit:           D0 = 0000 if OK
;                 D0 = BadTBMT if we timed out (-3110)
;                 A0,A1,D2 changed
;_______________________________________________________________

SendChar     _SUBR
             move     Ticks,D2              ; fail-safe counter
             add.l    #30,D2                ; bump by 1/2 second
@10          bsr.s    TestTBMT              ; look to see if we can send it
             bne.s    @20                   ; go if we can
             cmp.l    Ticks,D2              ; did we time out?
```

```
            bpl.s    @10                 ; go if not
            move     #-3110,D0           ; BadTBMT return code
            bra.s    @40

@20         bsr.s    SendSCC              ; else send it

@40         _SUBEND 'SENDCHAR'
;
; Check state of TBMT - sets CCR to state of TBMT
; Uses A0
;
TestTBMT     movem.l SCCRd,A0             ; point at the SCC
            IF       PortA THEN
            addq.l   #Actl,A0
            ENDIF
            btst     #TxEmptyBit,(a0)     ; is the TBMT set?
            rts                           ; return
            EJECT
;______________________________________________________________________
;
; TIntHnd -- this code catches the Tx Buffer Empty interrupts from
;       the SCC and tries to send another character.  If it could not
;       send a character, it clears the Tx Pending bit, so that the SCC
;       will not interrupt again.  Finally (in any case) it also resets
;       the highest interrupt under service (IUS) in the SCC to clear
;       the interrupt before returning.
;
; On entry, A0/A1 point to the SCC control read/write registers.
; Like a normal interrupt handler, it must preserve D4-D7 and A4-A7
;______________________________________________________________________

TIntHnd     move.l  MPPVars,a2           ; point at the MPP Variables
            _statcount XmitCount         ;

            sf      SentChar(A2)         ;
            bsr.s   TxNextCh             ; try to send another char

            tst.b   SentChar(A2)         ; did we?
            bne.s   TintIUS              ; go if so
            move.l  SCCWr,A1             ; otherwise reset TxPend
            IF      PortA THEN
            addq.l  #Actl,A1
            ENDIF
            move.b  #$28,(A1)
TIntIUS     bra     DoIUS                ; and reset the highest IUS

;______________________________________________________________________________
;
; TxNextCh -- try to send (in this order)
;           the next character of the segment, or
;           the next segment, or
;           the CRC, or
;           the trailing FrameChar.
;
; If a complete frame which was initiated by the device manager has
; been sent, we should jump thru IODone (asking the DevMgr for more
; to do).  Otherwise, (it was an IM or UR) we look to see if there
; is a frame from the DevMgr queued (in WDSptr).  If so, we start
; sending it, otherwise, we simply RTS.
;______________________________________________________________________________

TxNextCh    move.l  tWDSPtr(a2),D0       ; D0 -> WDS in progress
            beq.s   TxNextRTS            ; if nil, just exit (no message)
            tst.w   TxCount(A2)          ; is there more of the segment to send
            bne     LAPSend              ; if so, send next character

@5          move.l  D0,A0                ; otherwise, point at the WDS
            tst     (a0)                 ; check the next length
            beq.s   @10                  ; go if it's zero (end of the frame)
            move    (a0)+,TxCount(a2)    ; otherwise, update TxCount and
            move.l  (a0)+,LAPFetch(a2)   ;    and LAPFetch
            move.l  a0,tWDSptr(a2)       ; and update the tWDSPtr
            bra     LAPSend              ; and send it off
;
; Now send the CRC
;
@10         tst.b   nCRC(a2)             ; do we need to send a CRC?
            beq.s   @20                  ; go if not
            sf      nCRC(a2)             ; don't need one now
            move    outputCRC(a2),D0     ; get the two CRC bytes
            ror.w   #8,D0                ; swap them
```

(continued on next page)

ICs PROMPT DELIVERY!!!
SAME DAY SHIPPING (USUALLY)
QUANTITY ONE PRICES SHOWN for AUG. 23, 1987

OUTSIDE OKLAHOMA: NO SALES TAX

640K MOTHERBD UPGRADE: Zenith 150, IBM PC/XT, Compaq Portable & Plus; hp Vectra

DYNAMIC RAM			
1Mbit	1000Kx1	100 ns	$26.50
1Mbit	256Kx4	120 ns	32.00
51258	*256Kx1	100 ns	6.95
4464	64Kx4	150 ns	3.50
41256	256Kx1	80 ns	4.95
41256	256Kx1	100 ns	4.40
41256	256Kx1	120 ns	3.40
41256	256Kx1	150 ns	3.20
41264	2-PORT	120 ns	5.25
EPROM			
27512	64Kx8	200 ns	$11.25
27C256	32Kx8	250 ns	6.65
27256	32Kx8	250 ns	5.50
27128	16Kx8	250 ns	4.95
STATIC RAM			
43256L-12	32Kx8	120 ns	$11.95
5565PL-15	8Kx8	150 ns	3.25

* STATIC COLUMN DRAM FOR 386 COMPAQ

80387-16 $540.00
80287-8 $250.00
8087-2 $160.00

OPEN 6½ DAYS, 7:30 AM-10 PM: SHIP VIA FED-EX ON SAT.

SUNDAYS & HOLIDAYS: SHIPMENT OR DELIVERY, VIA U.S. EXPRESS MAIL

SAT DELIVERY INCLUDED ON FED-EX ORDERS RECEIVED BY:
Th: Std Air $4/1 lb
Fr: P-1 $10.50/2 lbs

MasterCard/VISA or UPS CASH COD
Factory New, Prime Parts μP∞
MICROPROCESSORS UNLIMITED, INC.
24,000 S. Peoria Ave.,
BEGGS, OK. 74421
(918) 267-4961

No minimum order. Please note that prices are subject to change. Shipping & insurance extra, & up to $1 for packing materials. **Orders received by 9 PM CST can usually be delivered the next morning, via Federal Express Standard Air @ $4.00, or guaranteed next day Priority One @ $10.50!** All parts guaranteed.

CIRCLE 105 ON READER SERVICE CARD

Quelo® 68000 Software Development Tools

First release 1983 · MOTOROLA compatible · produces ROMable code, S-records, extended TEK hex, UNIX COFF. Portable SOURCE CODE. Native and cross versions on: ATARI ST, AMIGA, Masscomp, Sun, Apollo, Charles River, VAX VMS and UNIX.

68020 Cross Assembler Package
Supports 68000, 68010, 68020, 68881 and 68851
For CP/M 68K and MS/PC DOS · $750

68000/68010 Cross Assembler Package
For CP/M 80, -86, -68K and MS/PC DOS - $595

68000 "C" Cross Compiler
For MS/PC DOS by Lattice, Inc. - $500

NEW ITEMS

68020 Disassembler
Supports 68000, 68010, 68020, 68881, 68851
For CP/M 68K and MS/PC Dos - $495/295
Amiga and Atari ST - $119/79. CRDS UNOS $995/595

68000/68010 Software Simulator
For MS/PC Dos by Big Bang Software, Inc. - $285. VAX - $1900

Call Patrick Adams *today:*
Site, Corporate, OEM licenses
COD, Visa, MasterCard

Quelo, Inc.
2464 33rd. West, Suite #173
Seattle, WA USA 98199
Phone 206/285-2528
Telex 910-333-8171

TM Quelo, Quelo, Inc. MS, Microsoft Corporation, CP/M, Digital Research

CIRCLE 377 ON READER SERVICE CARD

PLOT TEXT ON ANY GRAPHICS SCREEN!!

YES, we said ANY Graphics Screen, even VGA! FINALLY! Xgraf is a super set of smart low level assembly graphic routines that you call directly from Compiled BASIC. Xgraf replaces BASIC's confusing graphics statements with consistent, full featured calls specifically designed for the BASIC programmer.

FINALLY! Xgraf is only $99.00 + $4.00 S&H

We specialize in libraries and tools for Compiled BASIC. Our catalog features the FINALLY! Family of Products and other top flight tools.

Call: 1 800 423-3400
(9:00 AM to 8:00 PM EST)
PA & AK call (412) 782-0384

KOMPUTERWERK
851 Parkview Blvd.
Pittsburgh, PA 15215

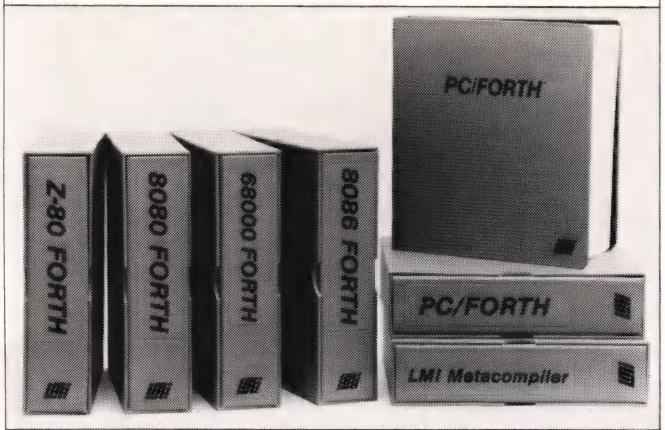
CIRCLE 388 ON READER SERVICE CARD

Listing One *(Listing continued, text begins on page 18.)*

```
            lea      CRCBuf(a2),a0       ; point at the CRC Tx Buffer
            move     D0,(a0)             ; save the CRC bytes
            move.l   a0,LAPFetch(a2)     ; and save the fetch pointer
            move     #2,TxCount(a2)      ; save the length, too
            bra      LAPSend             ; and send them off
;
; We've sent the CRC, now send the closing FrameChar
;
@20         tst.b    nFrmChr(a2)         ; do we need to send a FrameChar?
            beq.s    @30                 ; go if not
            sf       nFrmChr(a2)         ;
            moveq    #qFrmChar,D0        ; get $A5
            bra      SendSCC             ; send it and exit
;
; We've sent a full frame, now clean up
;
@30         clr.w    TxCount(a2)         ; clear the TxCount
            clr.l    tWDSPtr(a2)         ; clear the tWDSptr (no longer sending)
;
; Now decide whether to return, wakeup the Dev. Mgr, or start a queued frame
;
            tst.b    SendingIMUR(A2)     ; were we sending an IM or UR?
            beq.s    NotIMUR             ; go if not
            sf       SendingIMUR(A2)     ; well, we're not anymore
            move.l   qWDSptr(A2),D0      ; is there a queued frame?
            beq.s    TxNextRTS           ; go if not
            move.l   D0,A0
            bra      SendFrame           ; otherwise, start sending it
TxNextRTS   rts                          ; otherwise, return (RTS)
;
; We weren't sending IM/UR so we must have finished a msg from the
; device mgr.  Therefore, we should return to the Device Manager.
;
NotIMUR     clr.l    qWDSptr(A2)         ; clear out the WDS
            moveq    #0,D0               ; good return status
            bra      LAPWexit            ; and go thru LAPWrtRtn to IOdone
            EJECT
;________________________________________________________________________
;
; RandomWord - generate a random number
;
;   Call:
;       RandomSeed(A2) = seed
;
;   Return:
;       D0 = random number (CCR set to it)
;________________________________________________________________________

RandomWord  MOVE     RandomSeed(A2),D0   ; D0 = current seed
            MULU     #773,D0             ; Times 773
            ADDQ     #1,D0               ; Plus 1
            MOVE     D0,-(SP)            ; Save high byte on stack
            LSL      #8,D0               ; Put low byte into high byte
            MOVE.B   (SP)+,D0            ; And high byte into low byte
            MOVE     D0,RandomSeed(A2)   ; Set back in seed
            RTS
            EJECT
;______________________________________________________________________
;
; VBL handler - come here every VBLtimer ticks. Used to check for long
; output puases; if we stop for > 1 second, we expermientally send
; the next character.
;   A0 -> VBL queue element
;______________________________________________________________________

VBLHnd      MOVE     #VBLtimer,VBLCount(A0) ; Better re-init VBL count
            MOVE.L   MPPVars,A2             ; A2 -> local variables
;
; Have we sent an Xoff (did we set nXon)?  If so, try to send an Xon
;
            tst.b    nXon(A2)            ; do we need an Xon?
            beq.s    @20                 ; go if not
            bsr      TestTBMT            ; try to send it to the SCC
            beq.s    VBLHndRTS           ; quit if we couldn't send it
            moveq    #Xon,D0
            bsr      SendSCC             ; send an Xon
            sf       nXon(A2)            ; and clear the flag
            bra.s    VBLHndRTS           ; and quit
;
; Check for long pause during transmit
;
```

(continued on page 78)

TOTAL CONTROL with *LMI FORTH*™



For Programming Professionals:

an expanding family of compatible, high-performance, Forth-83 Standard compilers for microcomputers

For Development: Interactive Forth-83 Interpreter/Compilers

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

For Applications: Forth-83 Metacompiler

- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, and 6303
- No license fee or royalty for compiled applications

For Speed: CForth Application Compiler

- Translates "high-level" Forth into in-line, optimized machine code
- Can generate ROMable code

Support Services for registered users:

- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.

Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to: ***(213) 306-7412***

Overseas Distributors.
Germany: Forth-Systeme Angelika Flesch, Titisee-Neustadt, 7651-1665
UK: System Science Ltd., London, 01-248 0962
France: Micro-Sigma S.A.R.L., Paris, (1) 42.65.95.16
Japan: Southern Pacific Ltd., Yokohama, 045-314-9514
Australia: Wave-onic Associates, Wilson, W.A., (09) 451-2946

CIRCLE 205 ON READER SERVICE CARD

M Street Software

80386 Support!

SCRUTINY

Advanced symbolic debugger.

- Multi-language: compatible with Turbo Pascal, Microsoft Assembler, others.
- Multi-DOS: works with all MS-DOS/PC-DOS computers.
- Multi-level: debug at source level and machine level, separately or together.
- Multi-display: debug character-mode and graphics-mode programs, with movable debug windows.
- Multi-chip: support for 8086, 80186, 80286, 80386.
- Fast 80386 "memory breakpoints" (stop program when specified variable is accessed or modified).

Scrutiny/Master $99.95
for debugging Turbo Pascal, Microsoft Assembler, and other languages.

Scrutiny/Turbo Special price! $49.95
for debugging Turbo Pascal only.

VISA/MC AMEX accepted. In Texas please add sales tax. Outside of North America add $10 per item shipping.

M Street Software
5400 E. Mockingbird Lane Suite 114
Dallas, Texas 75206
214-827-4908

Information also available via our 24 hour 300/1200 modem: 214-669-1882.

CIRCLE 275 ON READER SERVICE CARD

Get Real.

Get real productive with REAL-TOOLS, a general purpose set of "C" development tools for UNIX™ and XENIX™.

Get Graphics Too! In addition to an advanced screen management system and superior windowing capabilities, REAL-TOOLS offers user-defined graphics for you to draw, save, recall, copy and animate symbols and panels.

So if you're developing applications for the real world — get real productive. Get graphics. Get REAL-TOOLS.

Real-Tools™

$99 Binary only. $549 Library source. $999 Complete source.

PCT

Pioneering Controls Technologies, Inc.
510 Bering Drive, Suite 300, Houston, Texas 77057
(713) 266-8649

™REAL-TOOLS is a trademark of Pioneering Controls Technologies, Inc.
™UNIX is a trademark of AT&T
™XENIX is a trademark of Microsoft Corporation.

CIRCLE 191 ON READER SERVICE CARD

Listing One *(Listing continued, text begins on page 18.)*

```
@20         tst.l   tWDSptr(A2)          ; do we have anything to send?
            beq.s   VBLHndRTS            ; return if not
            move.l  Ticks,D0
            sub.l   LastXmit(a2),D0      ; if (ticks - LastXmit) > 60 then
            cmp     #60,D0               ; let's try to send another char
            bmi.s   VBLHndRTS
            bsr     TestTBMT             ; is TBMT set (can we send another char?)
            beq.s   VBLHndRTS            ; go if not
            sf      RcvdXoff(a2)         ;
            _statcount XOFFTOcount
            MOVE    #SCCLockout,SR       ; exclude SCC interrupts (VIA priority < SCC)
            bsr     TxNextCh             ; otherwise, do another character
VBLHndRTS   rts                          ; this'll restore SR et al
            eject
;________________________________________________________________________
;
;   myPollProc -- AALAP PollProc addendum (predendum?):
;
;   The AALAP needs a bit of a PollProc, since it will lose characters
;   whenever the disk spins.  Of course, all good Macintosh programmers
;   know that the Printer Port (PortB) isn't polled by the disk driver
;   since there's just not enough horsepower to go around.
;
;   The PollProc is called by the disk driver to poll PortA.  We
;   execute a snippet of code before the real PollProc, and send an
;   Xoff to the other end if we're receiving or processing a message
;   while the disk is spinning.  Then we transfer to the real PollProc.
;
;   This routine preserves all regs except the SR.  It does this by
;   reserving a longword on the stack, and then stuffing the SavePS
;   value in it.  If it's zero, then there wasn't a PollProc, and we
;   pop that value off the stack and return to the disk driver.  If
;   that value wasn't zero, then the real PollProc's address will be
;   on the top of the stack, and we go there.  The disk driver's return
;   address will be left on the stack, allowing the PollProc to return
;   normally.
;
;   InpState and stillBusy must both be in the same word.  The
;   tst.w InpState(A2) below fails otherwise.
;________________________________________________________________________

myPollProc  subq    #4,A7                ; save space for a return adrs
            move.l  A2,-(SP)             ; and save A2
            move.l  MPPVars,A2           ; point at the MPP locals
            tst.b   nXon(A2)             ; have we already sent an Xoff?
            bne.s   myPPexit             ; go if so
            tst.w   InpState(A2)         ; are we receiving or processing a message?
            beq.s   myPPexit             ; go if not

            movem.l A0/A1/D0,-(SP)       ; save regs
@10         bsr     StashSCCch           ; grab a char from the SCC, save it
            bne.s   @10                  ; loop 'til it's empty
            statcount PPCount
            bsr     TestTBMT             ; is it OK to send the Xoff?
            beq.s   @30                  ; go if not
            moveq   #Xoff,D0
            bsr     SendSCC              ; send Xoff
            st      nXon(A2)             ; and remember we need Xon
            statcount PPXoffCnt
@30         movem.l  (SP)+,A0/A1/D0      ; restore the regs

myPPexit    move.l   SavePS(A2),4(SP)    ; move address onto stack (sets CC)
            movea.l (SP)+,A2             ; restore A2
            bne.s    @20                 ; go if PollProc adrs <> 0 (use it)
            addq.l   #4,SP               ; else pop the (nil) adrs
@20         rts                          ; and go there

            EJECT
;________________________________________________________________________
;
;  ExtIntHnd -- catch the External or Status Interrupts from the SCC
;
;  Checks for mouse interrupt, passes control if it is one, else resets
;  the external/status SCC interrupts.
;________________________________________________________________________

ExtIntHnd   btst    #DCDbit,D1           ; did the DCD bit change (mouse moved)
            beq.s   @10                  ; go if not
            move.l  MouseVector,A3       ; else, point at the mouse handler
            jmp     (A3)                 ; and go there
```

```
@10         move.b  #$10,(a1)              ; reset ext interrupts
            move.b  #$10,(a1)              ;     (twice)
            move.b  #ResetIUS,(a1)         ; Reset Highest IUS in SCC (to WR0)
            rts
            EJECT
;____________________________________________________________________________
;
; RIntHnd  - SCC receive interrupt handler
;
;   Called: A0 -> SCC control read register
;           A1 -> SCC control write register
;
; This code is structured differently from the ABLAP code, since
; the arrival rate of the chars is so much slower for AALAP.  Normal
; ABLAP routines call ReadPacket and ReadRest to get pieces or the rest
; of the frame as they arrive in real time.  With AALAP, the character
; arrival rate is so slow that we copy the entire frame into an
; interrupt-time buffer.
;
; When we receive a good frame, we then pass control to the appropriate
; protocol handler, which then makes calls on ReadPacket and ReadRest to
; dole out the characters as necessary.
;
; Like all Mac interrupt handlers, it must preserve D4-D7 and A4-A7.
; and return with a RTS instruction.
;
; Since the default DDP socket listener is quite slow (3-4 msec to process
; a newly received message) we set up a buffer to contain characters
; which arrive during the time the socket listener is in control.  We
; set a flag (stillBusy) to indicate that we're still busy, and save the
; chars in BusyBuf.
;____________________________________________________________________________

SpIntHnd
RIntHnd     move.l  MPPVars,A2             ; A2 -> driver variables
            _statcount RcvIntCount         ; remember the number of Rcv
interrupts

RIntHnd10   bsr     NextChar               ; handle next char (from BusyBuf or
SCC)
            beq     RIntRTS                ; quit if no data

            and     #$00FF,D0              ; use only eight bits
            move.w  D0,LastRxCh(a2)        ; remember the char
;
; Check for flow control from other side
;
@15         move.b  D0,D1                  ; check for either parity Xon/Xoff
            and.b   #$7F,D1
            cmp.b   #Xoff,D1               ; is it a control-S?
            bne.s   @20                    ; go if not
            _statcount XOFFcount           ; count it
            st      rcvdXoff(a2)           ; and remember we received Xoff
            bra.s   RIntHnd10              ; loop for another char

@20         cmp.b   #Xon,D1                ; or is it a control-Q?
            bne.s   @30                    ; go if not
            _statcount XONcount
            sf      rcvdXoff(a2)
            bsr     TestTBMT               ; is the tx empty?
            beq.s   RIntHnd10              ; loop if not
            bsr     TxNextCh               ; otherwise, start up Tx side again
            bra.s   RIntHnd10              ; loop for another char
;
; Watch out for framing characters
;
@30         cmp.b   #FrameChar,D0          ; is it a framing character?
            beq.s   GotFrmCh               ;     go if so
            tst.b   InpState(a2)           ; are we in a frame?
            beq.s   RintHnd10              ;     loop for another char
            EJECT
;
; Maybe this is a data char -- check the frame length
;
            cmp     #MaxLAPFrmLen,rcvdlen(a2) ; is the frame too long?
            bls.s   @50                    ; go if it's OK
            _statcount LongFrame           ; remember the long frame
            sf      InpState(a2)           ; go idle
            bra.s   RIntHnd10              ; loop for another char
;
; We have a real char -- un-escape it
```

(continued on next page)

68020
UNIX* COMPATIBLE MULTIUSER SYSTEM

The MPULSE Model 20

Multi-Processor Design:

Separating application processing from I/O is a well understood goal in multiuser supermicrocomputer design. As the real UNIX system bottleneck is disk I/O bandwidth, not raw processor speed, LPC has spent a great deal of time and development effort in optimizing disk I/O throughput. The result is a disk I/O subsystem unparalleled in the under $20,000 microcomputer class.

A fully asynchronous, high speed DMA channel links the MC68020 to the dual MC68000 I/O processors. A full 2 Mbytes of I/O processor memory is available for disk caching. The disk cache utilizes a least recently used, delayed write algorithm to achieve hit rates exceeding 90%.

In addition to disk caching, LPC has extended the conventional UNIX caching mechanism with a new virtual caching technique, implemented in the kernel itself. The Dynamic Kernel Cache (DKC) distributes available memory between user processes and the internal UNIX cache. This dynamic allocation technique allows a much more efficient use of main memory with cache efficiency increased to over 80%, much higher than the conventional UNIX caching mechanism.

Operating System:

The MPULSE Model 20 hosts LPC's System V operating system, derived from the industry standard UNIX System V. The MPULSE System V port is tailored to complement the MPULSE hardware while retaining compatibility at all levels with the generic UNIX System V operating system. Areas of optimization and additional features include:

- Full demand-paged virtual memory
- Kernel portions of the terminal and mass storage I/O disciplines are distributed to the appropriate I/O processors
- Berkeley enhancements
- Dynamically distributed ramdisks
- On-line Winchester disk bad block replacement
- On-line device configuration
- True multisector I/O for streaming tape operation
- Support for implementing concurrent operating systems
- General purpose user-accessible SCSI device driver
- Automatic bi-directional modem support
- MWindows windowing capability

Base System Includes:

- 16 MHz MC68020 host processor
- DUAL 12 MHz MC68000 I/O sub-system
- 4 Mbytes main memory
- 2 Mbytes of dedicated disk cache memory
- Demand-Paged Virtual Memory
- Sophisticated LRU caching algorithm
- Dynamic Kernel Cache (DKC)
- MWindows
- 50 MByte hard disk expandable to 0.5 gigabytes
- 800 Kbyte floppy drive
- 60 Mbyte cassette tape backup
- 8 serial ports expandable to 40 serial ports
- LPC System V derived from UNIX System V
- Berkeley Enhancements
- NCR Tower object code compatibility

Base System Price:

$5995.00
Call (214) 340-5172

Warranty:
90 day (extended warranty available)
15 day money back evaluation period

LPC Logic Process Corporation
10355 Brockwood Road Dallas, Texas 75238

DKC and MWindows are trademarks of LPC.
UNIX is a trademark of AT&T.
Tower is a trademark of NCR.

CIRCLE 169 ON READER SERVICE CARD

Cross Assemblers
Universal Linker
Powerful Librarian

PC/MS DOS, micro VAX,
VAX VMS, VAX UNIX/ULTRIX

- Targeting over 30 Microprocessors
- Version 2.1 is FAST
- Powerful Macros
- Absolute or Relocatable Code
- Compatible with all Assemblers
- Conditional Assembly
- $295 up for Complete Packages

Next Month
Microcontroller
C cross compilers

Enertec inc.
19 Jenkins Ave.
Lansdale, PA 19446 U.S.A.
telephone: 215-362-0966
telex: 4948709 ENERTEC

CIRCLE 346 ON READER SERVICE CARD

Changing Your Address?

To change your address, attach your address label from the cover of the magazine to this coupon and indicate your new address below.

Name

Address

Apt. # State

City

Zip

Mail to:

Dr. Dobb's Journal,
PO Box 27809,
San Diego, CA 92128

Listing One *(Listing continued, text begins on page 18.)*

```
;
@50         cmp.b   #DLE,D0             ; is it a DLE?
            bne.s   @90                 ; go if not
            st      EscIn(a2)           ; remember we've seen an escape
            bra.s   RIntHnd10
;
; This is a data char -- complete any escaping, accumulate the CRC
;
@90         tst.b   EscIn(a2)           ; should we escape it?
            beq.s   @100                ; go if not
            eor     #$40,D0             ; xor with $40
            sf      EscIn(a2)           ; and clear the escape flag

; now we've got a good char
@100        lea     inputCRC(a2),a3     ; point at the CRC accumulator
            bsr     NextCRC             ; update the CRC accum using byte in
D0
            move.l  LAPStash(a2),a0     ; point at the next free char in
buffer
            move.b  D0,(a0)+            ; save the char in the buffer, bump the
pointer
            addq    #1,rcvdlen(a2)      ; increment the bytes-read counter
            cmp     #3,rcvdlen(a2)      ; have we read in exactly three chars?
            bne.s   @110                ; go if not
            move.l  LAPInBuf(a2),a0     ; otherwise point at the LAPInBuf
@110        move.l  a0,LAPStash(a2)     ; and update the pointer
            bra     RIntHnd10           ; loop for another char

RIntRTS     bra     DoIUS               ; reset Highest IUS and return

;
; We've discovered a FrameChar -- check if we're done or just starting
;
GotFrmCh    tst.b   InpState(a2)        ; are we in a frame?
            beq.s   FrmStart            ;  go if not (we will be)

FrmEnd      cmp     #2,rcvdlen(a2)      ; found closing char
            bhi.s   CheckCRC            ;    go if frame is long enough
            _statcount ShortFrame       ; else, flag that we got a short frame
                                        ;    and fall into FrameStrt
;
; We're in a frame now!
;
FrmStart    lea     toRHA(a2),a3        ; a3 -> RHA (holds 1st 5 bytes)
            move.b  sysLAPAddr(a2),(a3)+ ; copy the node number
            move.b  sysABridge(a2),(a3)+ ; and the bridge address
            move.l  a3,LAPStash(a2)     ; remember where next byte goes
            st      InpState(a2)        ; change the InpState to in_msg
            sf      EscIn(a2)           ; and we're not escaping data
            clr     InputCRC(a2)        ; no CRC yet
            clr     rcvdlen(a2)         ; no data, either
            bra.s   RIntRTS
            EJECT
;
; We received a complete frame -- check the CRC
;
CheckCRC
            sf      InpState(a2)        ; we're not in a frame now
            tst     InputCRC(a2)        ; is the CRC zero?
            beq.s   LAPDemux            ; go if it is OK
            _statcount CRCCount         ; save the statistic
            bra.s   RIntRTS             ; and exit
;
; Come here on receipt of a good frame.  We've cleared the InpState
; to indicate we're out of a frame.
;

LAPDemux    _statcount FrmCount         ; log another good frame
            move.l  Ticks,LastRcv(a2)   ; remember this frame's arrival time
            lea     2+toRHA(a2),a3      ; a3 -> LAP type byte
            MOVE.B  (A3)+,D0            ; Get the LAPtype, bump pointer
            tst.b   D0
            BMI     LAPIn               ; If minus, it's a LAP packet

;
; Got a data packet - look for a protocol handler
;
            tst.b   AALAPup(a2)         ; but first, is the AALAP up?
            beq.s   @60                 ; go if it's not up
            MOVEQ   #(LAPTblSz-1),D2    ; D2 = index into active protocols list
@30         CMP.B   Protocols(A2,D2),D0 ; Match?
```

```
        DBEQ    D2,@30                ; (If none, D2 is negative - 3.1F)
        LSL.W   #2,D2                 ; Make D2 a longword index into Handlers
;
; Got a protocol handler -- Compute the desired length of the message in D1
;
        move.b  (a3)+,D1              ; Get MSByte of the length into D1
        and     #3,D1                 ; mask for two lsbits
        LSL     #8,D1                 ; Move to proper position
        MOVE.B  (a3)+,D1              ; D1 = total length
        move    rcvdlen(a2),D0        ; D0 = total chars received (DDP + LAP
+ CRC)
        subq    #3,D0                 ; disregard LAP type and CRC
        cmp     D1,D0                 ; are they equal?
        beq.s   @40                   ; go if so
        _statcount LenErrCnt          ; save the stats
        bra     RIntRTS               ; and exit

@40     SUBQ    #2,D1                 ; Subtract 2 for length bytes
        move    d1,RcvdLen(a2)        ; and remember the number of unread chars
        EJECT
;_______________________________________________________________________
;
;       At this point, Handlers(A2,D2) points to the address of the protocol
;       handler for this packet's protocol (or D2 is negative if there is
;       none -- 3.1F). JMP to it with the following:
;
;       A0,A1 = SCC read/write addressses
;       A2 = ptr to driver locals
;       A3 = ptr into the RHA (first 5 bytes loaded)
;       A4 will be the address of our read packet routine
;       A5 will be saved for handler's usage (until packet's all in or error)
;       D1 = length of packet still left to read (from header)
;
;       The protocol handler must obey the following conventions:
;
;       1) It must preserve, across the call, A0-A2, A4 and D1
;       2) A6 and D4-D7 must be saved and restored if used.
;       3) It must JSR to the routine at (A4) or 2(A4) with registers as defined
;          there, for the purpose of reading more of the packet and eventually
;          resetting the SCC for the next interrupt.
;_______________________________________________________________________

        TST     D2                    ; Is there a protocol handler? (3.1F)
        BMI.S   @60                   ; Branch if not
        bsr     DoIUS                 ; reset Highest IUS
        MOVEM.L A4/A5,SaveA45(A2)     ; Save A4 and A5 (may be free time now)
        move.l  LAPInBuf(a2),a4       ; point at the next char of the msg
        move.l  A4,LAPStash(a2)       ;    (we can snatch A4 for a few instrs)
        MOVE.L  Handlers(A2,D2),A5    ; A5 -> protocol handler
        LEA     ReadPacket,A4         ; A4 -> ReadPacket
        st      stillBusy(a2)         ; remember we're processing a frame
        move.w  VSCCEnable(A2),SR     ; re-enable so we can catch more chars (!)

        JSR     (A5)                  ; Call the protocol handler

        move.l  MPPVars,A2            ; point at our variables
        cmpa.l  SaveA45(A2),A4        ; paranoia land -- make sure they've left
        bne.s   @45                   ;    things as they should be
        cmpa.l  (SaveA45+4)(A2),A5
        beq.s   @50
@45     pea     BadA4A5
        DC.W    $ABFF                 ; print the text (in lieu of $A9FF)
@50     sf      stillBusy(A2)         ; and now we're not in a frame
        rts                           ; exit the interrupt handler
;
; No handler, just log the error
;
@60     _StatCount NoHandCnt          ; Count packets without a handler
        bra     RIntRTS               ; and exit
BadA4A5 DC.B    17                    ; debugging only
        DC.B    'AALAP - Bad A4/A5'
        align 2
        EJECT
;_______________________________________________________________________
;
;   NextChar -- Handle the next char
;
;   This routine does two things:  If we're awaiting a full message, then
;   it gets the next character.  That char may have arrived from the SCC,
;   or it may be a char left in the BusyBuf. (Chars in the BusyBuf take
;   precedence.)
```

(continued on next page)

NOW FOR IBM PC, XT, AT, PS2 AND TRS-80 MODELS 1, 3, 4, 4P

The Gifted Computer

1. Buy **MMSFORTH** before year's end, to let your computer work harder and faster.
2. Then MMS will reward it (and you) with the **MMSFORTH GAMES DISK**, a $39.95 value which we'll add on at **no additional charge**!

MMSFORTH is the unusually smooth and complete Forth system with the great support. Many programmers report **four to ten times greater productivity** with this outstanding system, and MMS provides **advanced applications programs** in Forth for use by beginners and for custom modifications. Unlike many Forths on the market, MMSFORTH gives you a rich set of the instructions, editing and debugging tools that professional programmers want. The licensed user gets **continuing, free phone tips** and a **MMSFORTH Newsletter** is available.

The **MMSFORTH GAMES DISK** includes arcade games (BREAKFORTH, CRASH-FORTH and, for TRS-80, FREEWAY), board games (OTHELLO and TIC-TAC-FORTH), and a top-notch CRYPTO-QUOTE HELPER with a data file of coded messages and the ability to encode your own. All of these come with **Forth source code**, for a valuable and enjoyable demonstration of Forth programming techniques.

Hurry, and the GAMES DISK will be our free gift to you. Our **brochure** is free, too, and our knowledgeable staff is ready to answer your questions. **Write. Better yet, call 617/653-6136.**

MMSFORTH

and a free gift!

GREAT FORTH:
MMSFORTH V2.4 .... $179.95*
The one you've read about in FORTH: A TEXT & REFERENCE. Available for IBM PC/XT/AT/PS2 etc., and TRS-80 M.1, 3 and 4

GREAT MMSFORTH OPTIONS:
FORTHWRITE $99.95*
FORTHCOM 49.95
DATAHANDLER 59.95
DATAHANDLER-PLUS* 99.95
EXPERT-2 .... 69.95
UTILITIES 49.95
*Single-computer, single-user prices; corporate site licenses from $1,000 additional. 3½" format, add $5/disk; Tandy 1000, add $20. Add S/H, plus 5% tax on Mass. orders. DH+ not avail. for TRS-80s.

GREAT FORTH SUPPORT:
Free user tips, MMSFORTH Newsletter, consulting on hardware selection, staff training, and programming assignments large or small.

GREAT FORTH BOOKS:
FORTH: A TEXT & REF. $21.95*
THINKING FORTH 16.95
Many others in stock.

MILLER MICROCOMPUTER SERVICES
61 Lake Shore Road, Natick, MA 01760
(617/653-6136, 9 am - 9 pm)

CIRCLE 263 ON READER SERVICE CARD

Listing One *(Listing continued, text begins on page 18.)*

```
;
;   If we're still processing the previous message (stillBusy set true),
;   then all characters which arrive will be placed in BusyBuf, and the
;   associated pointers updated.  (Note: myPollProc also inserts data
;   into the BusyBuf, but it doesn't set stillBusy.)
;
;   Uses    A0,A1,D0
;   Assumes A2 -> MPPVars
;
;   Returns Z if no character
;           NZ if char present (char is in 8 lsbits of D0)
;_______________________________________________________________________
NextChar    _SUBR
            tst.b   stillBusy(A2)       ; are we still processing the prev.
frame?
            bne.s   @30                 ; go if we are

            bsr.s   GetBusyChar         ; else, look for a char from BusyBuf
            bne.s   @50                 ; quit if we got one
            bsr.s   GetSCCchar          ; else check the SCC
            bra.s   @50                 ; and quit

@30         bsr.s   StashSCCch          ; stash a char from SCC into BusyBuf
            bne.s   @30                 ; go back and look for more

@50         _SUBEND 'NEXTCHAR'
            _assumeEq BusyStash,BusyBuf+16 ; otherwise cmpa.l A0,A1 (above)
fails

GetBusyChar _SUBR                       ; get a char from the BusyBuf
            move.l  BusyFetch(A2),D0    ; get the fetch pointer
            cmp.l   BusyStash(A2),D0    ; is it the same as the stash pointer
            bne.s   @10                 ; go if not (more chars to do)
            lea     BusyBuf(a2),a0      ; point at the busy buffer
            move.l  a0,BusyStash(A2)    ; and save it in the BusyStash
            move.l  a0,BusyFetch(A2)    ; and BusyFetch
            moveq   #0,D0               ; clear the CC
            bra.s   @20
@10         move.l  D0,A0               ; there's still more to take
```


The
Heap Expander

- dynamically allocates data storage space in expanded memory
- simple interface
- up to 8 megabytes of heap space with appropriate hardware
- libraries and source code for:
  —Microsoft C, Lattice C, Turbo C, Mark Williams C, and others
  — Turbo Pascal
  — Logitech Modula-2
- requires IBM PC, XT, AT, or close compatible with LIM-standard expanded memory and MS—DOS or PC—DOS ver. 2.0 or above
- MC/VISA/COD call
  1-800-248-1045 x100 (US)
  1-800-952-5560 x100 (Idaho)

$59.95*

The Tool Makers

P.O. Box 8976
Moscow, Idaho 83843
(208) 883-4979

*Idaho residents add 5% sales tax

CIRCLE 319 ON READER SERVICE CARD

New!

386|DEBUG

- A symbolic debugger for 80386 32-bit *protected mode* programs which run under Phar Lap's 386|DOS-Extender™
  - Breakpoints, data watchpoints, and built-in disassembler
- Fully compatible with Phar Lap's **386|ASM/LINK,** the MetaWare 80386 **High C™** and **Professional Pascal™** compilers, and the Green Hills 80386 **Fortran** compiler
- Runs on all DOS-based PCs equipped with an 80386 CPU, including the Compaq® DESKPRO 386™, the IBM®PS/2™ Model 80, and most accelerator cards, including the Intel Inboard™ 386/AT
- $195—Available *today*

(617) 661-1510

Phar Lap Software, Inc.
60 Aberdeen Ave.
Cambridge, MA 02138

PHAR LAP

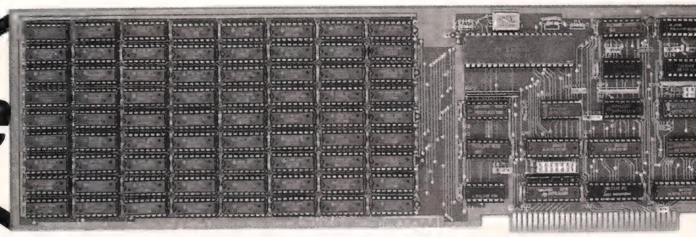
"The 80386 Software Experts"

CIRCLE 343 ON READER SERVICE CARD

```
            move.b  (A0)+,D0            ; get the byte
            move.l  A0,BusyFetch(A2)    ; update the pointer
            or.w    #$100,D0            ; make CC <> Z (must preserve 8
lsbits)
@20         _SUBEND 'GETBUSYC'
            EJECT
;____________________________________________________________________________
;
; GetSCCchar and StashSCCch both are called by RintHnd and myPollProc
; BOTH ROUTINES MAY ONLY USE A0, A1, AND D0!!!!!  (A2 will -> MPPVars)
;
;   GetSCCchar looks at RCA on the proper channel, and returns the char
;   in D0 if there was one (with CC set <> Z); else it returns CC = Z.
;____________________________________________________________________________
GetSCCchar  movem.l SCCRd,A0/A1         ; forces A0/A1 to point at SCC
            IF      PortA THEN
            addq.l  #Actl,A0
            addq.l  #Actl,A1
            ENDIF
            btst    #RCAbit,(A0)        ; is there a char?
            beq.s   @20                 ; go if not
            move.b  #1,(a1)             ; point at the error bits from RR1
            nop
            move.b  (a0),D0             ; get them (Overrun,Framing) in D0
            and     #$70,D0             ; any error bits?
            beq.s   @10                 ; go if not
            move.b  #ResetErr,(a1)      ; else send Error Reset to WR0
            nop
            move.b  #1,(a1)             ; point at WR1
            nop
            move.b  #$13,(a1)           ; and set up for int on all rx chars
            nop
            _statcount OVRcount         ; count 'em
@10         move.b  SCCData(a0),D0      ; and get the data (EVEN IF ERROR!)
            or.w    #$100,D0            ; set the SR (to NZ -- there's a char)
@20         rts
;
;   StashSCCch -- take a char from SCC, save in BusyBuf if there's space
;    Return Z if no char or no space; NZ otherwise
;
StashSCCch  bsr.s   GetSCCchar          ; look for a char in the SCC
```

(continued on next page)

SemiDisk® has an *attractive* personality.



"A while back I got a SemiDisk to help me with my database work. A SemiDisk is like a RAM-disk only a whole lot better. It doesn't sit in my main or EMS memory, and, using the Battery Backup, it's like permanent storage.

"That SemiDisk makes light work of the jobs that were sending my hard disk to an early grave. And SemiDisk has no head to crash; no moving parts to wear out. With all the time it saves me, I figure it paid for itself in just a couple of months.

"Then I heard programs like Microsoft Windows could use my SemiDisk for temporary files instead of using EMS. So I moved them over to the SemiDisk, too. The quiet speed of it is almost elegant!

"My boss wanted to try my SemiDisk on the company LAN server, but I told him to get his own. A couple of days later, he was wearing a grin as big as mine. I guess he likes his SemiDisk too.

"One morning I booted up my computer and there was my word processor waiting for me on the SemiDisk. I swear I didn't put it there! After I tried it, I knew it was there to stay.

"Meanwhile, I've found a new use for my hard disk, too. It's great for backing up my SemiDisk!"

I/O mapped SemiDisk goes in standard PC, XT or AT expansion slot. Priced at just $495 for 512K, $795 for 2Mb. Battery Backup $130. Up to 8Mb per drive. Call or write for further information or to place an order.

SemiDisk

End the waiting.

SemiDisk Systems, Inc.
P.O. Box GG
Beaverton, OR 97075
(503) 626-3104

VISA MasterCard

CIRCLE 85 ON READER SERVICE CARD

CQL

SQL Compatible Query System adaptable to any operating environment.

CQL Query System. A subset of the Structured English Query Language (SEQUEL, or SQL) developed by IBM. Linked files, stored views, and nested queries result in a complete query capability. File system interaction isolated in an interface module. Extensive documentation guides user development of interfaces to other record oriented file handlers.

Portable Application Support System

Portable Windowing System. Hardware independent windowing system with borders, attributes, horizontal and vertical scrolling. User can construct interface file for any hardware. Interfaces provided for PC/XT/AT (screen memory interface and BIOS only interface), MS-DOS generic (using ANSI.SYS), Xenix (both with and without using the curses interface), and C-library (no attributes).

Screen I/O, Report, and Form Generation Systems. Field level interface between application programs, the Query System, and the file system. Complete input/output formatting and control, automatic scrolling on screens and automatic pagination on forms, process intervention points. Seven field types: 8-bit unsigned binary, 16 bit signed binary, 16 bit unsigned binary, 32 bit signed binary, monetary (based on 32 bit binary), string, and date.

Including Source Code

$395.00

File System interfaces include C-tree and BTRIEVE.

HARDWARE AND FILE SYSTEM INDEPENDENT

MACHINE INDEPENDENT SOFTWARE CORPORATION

1415 NORTHGATE SQUARE #21D
RESTON, VA 22090

VISA/Master Charge accepted
(703) 435-0413

*C-tree is a trademark of FairCom

IBM, SEQUEL, PC, XT, AT are trademarks of IBM Corp. MS-DOS and Xenix are trademarks of Microsoft Corp.

CQL and the CQL logo are trademarks of Kurtzberg Computer Systems.

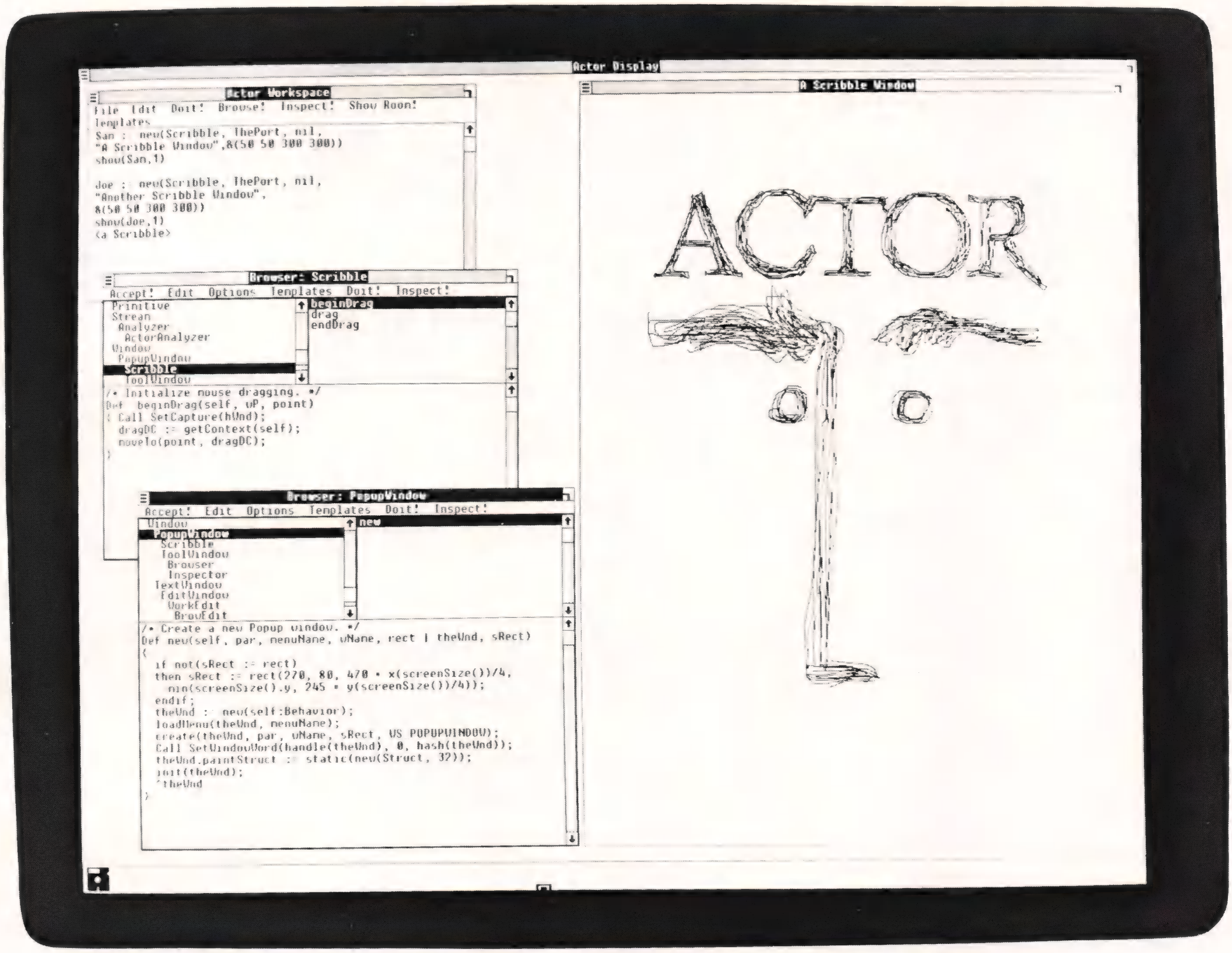
CIRCLE 294 ON READER SERVICE CARD

Listing One *(Listing continued, text begins on page 18.)*

```
            beq.s    @50                    ; go if none
            lea      BusyStash(a2),A1       ; point at the BusyStash pointer
            move.l   (a1),A0                ; and get it
            cmpa.l   A0,A1                  ; will this be too many chars?
            beq.s    @50                    ; yes, simply exit (and ignore the char)
            move.b   D0,(a0)+               ; save the char, and bump the pointer
            move.l   A0,BusyStash(A2)       ; and update the pointer
            or.w     #$100,D0               ; set the CC <> Z ('cause we took one )
@50         rts                             ; and return
            EJECT
;_______________________________________________________________
;
; DoIUS  -- reset Highest IUS
;_______________________________________________________________

DoIUS       _SUBR
            move.l   SCCWr,A1               ; point at the SCC write regs
            IF       PortA THEN
            addq.l   #Actl,A1
            ENDIF
            move.b   #ResetIUS,(a1)         ; Reset Highest IUS in SCC (to WR0)
            _SUBEND  'DOIUS   '
            EJECT
;_______________________________________________________________
;
; LAPIn - it's a LAP control packet.
;
;    D0 = LAP type
;    A3 -> remainder of the frame
;    Note:  for IM/UR frames, the net number (2 bytes) is at (a3),
;           but the node number (1 byte) is the first byte in LAPInBuf
;_______________________________________________________________
;
; Check for IM
;
LAPIn       move     (a3),D1                ; D1 = Net number (a3 sb even)
            move.l   LAPInBuf(a2),A0        ; point at first char in input buf
            move.b   (a0),D2                ; D2 = node number
            cmp.b    #lapIM,D0              ; is it an IM?
            bne.s    @60                    ; go if not
            move     D2,D0                  ; D0 = node number
            sf       RcvdXoff(A2)           ; so we can start sending
            bsr.s    CheckIM                ; figure out the net and node to send
            bsr.s    SendIMUR               ; send 'em
            bra.s    @80
;
; Check for UR
;
@60         cmp.b    #lapUR,D0              ; is it a UR?
            bne.s    @80                    ; go if not
            move     D2,D0                  ; D0 = Node number (D1 = Net number)
            bsr.s    CheckUR                ; check these values, return <> 0 if OK
            sne      AALAPup(a2)            ; if non-zero, then we're up
@80         rts                             ; and return

            _AssumeEq    lapENQ,$81         ; (1)
            _AssumeEq    lapRTS,lapENQ+3    ; (2)
            _AssumeEq    lapCTS,lapRTS+1    ; (3)
            EJECT
;_______________________________________________________________
;
; CheckIM -- check the received IM frame, compute UR response
;
; Entry:  D0 = their node number
;         D1 = their network number
; Exit:   D0,D1 = node, net number for the UR
;         D2 = qlapUR
;         Changes A0,A1,A3, D0-D3
;_______________________________________________________________

CheckIM     move.l   #0,A0                  ; return nil sometimes
            move.w   SysNetNum(a2),D2       ; D2 = our Net number
            beq.s    @10                    ; go if so -- check the node numbers
            move     D2,D1                  ; else, use our net number
@10         move.b   SysLAPAddr(a2),D3      ; D3 = our node number
@15         tst.b    D0                     ; while (theirnode <> 0)
            beq.s    @18                    ; &      (theirnode <> mynode)
            cmp.b    D3,D0                  ; have we both chosen the same value?
            bne.s    @20                    ; go if not -- return their value
@18         bsr      RandomWord             ; choose a random value
            and      #$7F,D0                ; mask to 7 bits
```

(continued on page 86)

HOW TO WRITE A WINDOWS APPLICATION IN TEN MINUTES.



Actor™ is a new language that combines Microsoft® Windows with object-oriented programming. This means you can produce mouse and window applications very quickly.

For example, we created a simple "paint" program, and used it to draw the Actor logo you see on the screen. The whole program only took ten lines and ten minutes. Part of it is in the middle window on the left.

Above, you see the commands that initialized the paint window and made it appear on the screen. Below, some code that's built into Actor, specifying window behavior. Through a process known as "inheritance," it's called into play automatically.

Try programming in this new way, and you'll never go back.

Find out about Actor.
Call The Whitewater Group®, (312) 491-2370.

Technology Innovation Center
906 University Place, Evanston, IL 60201

MICROSOFT WINDOWS

CIRCLE 282 ON READER SERVICE CARD

MINIMIZE TURBO
PASCAL DEBUG TIME

Tmark allows Turbo to continue compiling after an error is found without returning to line one!

Picture this:

You have just added a new routine at line 2,000 to an existing program. The routine contains 5 simple errors. You are going to have to compile those first 2,000 lines 5 times in order to find all the errors. That's 10,000 line of code!

But now you compile that program with Tmark installed. You will still have to compile the first 2,000 lines in order to find the first error. However, when you fix that error Turbo will continue compiling where you left off, not back at line one. You will be able to skip from error to error with no recompiling!

Tmark creates image files on disk to save and restore Turbo's state during a compilation. Saves are made automatically before compiler errors or at lines designated with a {tmark} comment. A window pops up to let you select which image to use to resume the compilation.

Tmark dramatically reduces debug time. Once you try it you will never want to give it up! $80 + $2 s/h, Visa/MC

TANGENT DESIGNS
PO Box 896, Lake Forest, IL 60045
(800) 356-2750, (312) 295-0030

CIRCLE 364 ON READER SERVICE CARD

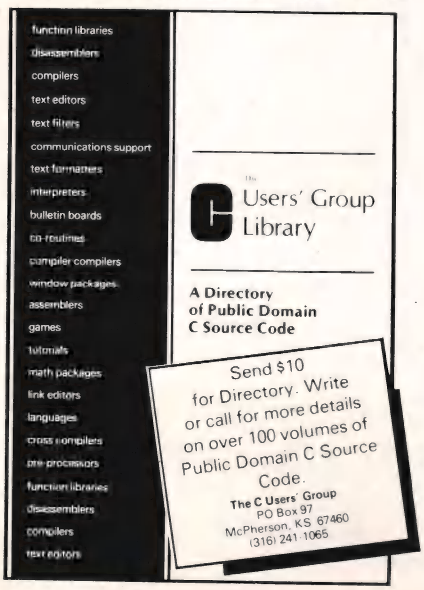
NEW! TLIB™ 4.0
SOURCE CODE CONTROL
The best keeps getting better!

- Ver. 3 reviewed in Sept 87 PC Tech Journal!
- The **fastest, most powerful** system is now even faster!
- *Many new features!* Keyword support - inserts date, version, history, etc. into source code. Extended wildcard and list-of-file support; can create lists by scanning source code for includes. Branching support, for multiple development lines. Can merge (reconcile) multiple simultaneous changes.
- Keep **all** versions of a source code file in **one** compact library. Synchronized control of multiple related source files.
- *LAN-compatible!* Share libraries with all popular networks. Check-in/out locking for multi-programmer projects.
- *Designed for the future!* Ideal for use with WORM optical disks, like the new IBM 3363, since libraries are appended, not replaced, when you add new versions.
- Includes a copy of Landon Dyer's excellent public domain **MAKE** utility (with source code for DOS & VAX/VMS).
- **Plus:** File compare utility. Virtually unlimited source file size. Date, comments with each version. Configurable user interface, and many configurable options, like: read-only libraries, automatic tab/blank conversion, more.

PC/MS-DOS 2.x & 3.x ***Just $99.95** + $3 s/h* Visa/MC

BURTON SYSTEMS SOFTWARE
P. O. Box 4156, Cary, NC 27519-4156
(919) 469-3068

CIRCLE 212 ON READER SERVICE CARD



CIRCLE 181 ON READER SERVICE CARD

ASYNC APPLETALK

Listing One *(Listing continued, text begins on page 18.)*

```
            bra.s     @15                   ; loop to insure they're different
@20         move.b    D0,sysABridge(a2)     ; remember their node number
            moveq     #qlapUR,D2            ; D2 = LAP type
            rts

            EJECT
;_______________________________________________________________________
;
; CheckUR -- check the received UR frame
;
; Entry:  D0 = node number
;         D1 = network number
; Exit:   D0 =  0 if net/node didn't match
;            <> 0 if they matched right off
;
;
;_______________________________________________________________________

CheckUR     SUBR
            cmp       SysNetNum(a2),D1      ; Network numbers match?
            bne.s     @10                   ; go if not
            cmp.b     SysLAPAddr(a2),D0     ; Node number match?
            bne.s     @10                   ; go if not
            moveq     #-1,D0                ; make D0 non-zero (it's OK)
            bra.s     CkURRTS               ; and exit
@10         tst       SysNetNum(a2)         ; is our network number 0000?
            bne.s     @50                   ; go if not (we cannot resolve this)
            move      D1,SysNetNum(a2)      ; save their Net/Node suggestions
            move.b    D0,SysLAPAddr(a2)
            bra.s     @60

@50         st        AALAPstuck(a2)        ; we're really bad off -- NNNN conflict
@60         clr       D0                    ; we didn't match

CkURRTS     _SUBEND 'CHECKUR '
            EJECT
;_________________________________________________________________
;
; SendIMUR - This routine fills and sends an IM or UR frame.  This is
;    a bit dicey, since a UR may be required as a result of receiving
;    an IM.  Since it's difficult to abort a frame already in progress,
;    we finesse the problem by not sending the IM/UR frame.  Here's why
;    it works:
;
;    A UR response is only necessary in two cases:
;      a) we're trying to bring the link up, and the other guy said "IM";
;      b) he hasn't heard from us, and he wants to make sure we're here.
;
;    For a), we shouldn't be talking, but he'll ask again anyway;
;    for b), the IM is trying to force us to send a good frame.
;            If the frame in transit makes it, OK.  If not, he'll
;            still ask again.
;
; Entry:     A0 -> master pointer of this hdlblk
;            A2 -> MPPVars
;            D0 = node number
;            D1 = Net number
;            D2 = LAP type
; Exit:      A0,A1,A3,D0-D3 changed
;_______________________________________________________________________

SendIMUR    _SUBR
            tst.l     tWDSptr(A2)           ; are we sending?
            bne.s     SndIMUR1              ; yes, just return
            lea       IMURbuf+1(A2),A1      ; A1 points at IMURbuf (odd adrs)
            move.b    D2,2(a1)              ; save the LAPtype (IM or UR)
            move.w    D1,3(a1)              ; and the Net number
            move.b    D0,5(a1)              ; and the Node number

            lea       IMURwds(A2),A0        ; A0 points at the WDS
            move.w    #6,(A0)               ; save the length
            move.l    A1,2(A0)              ; and the pointer to the data
            clr.w     6(A0)                 ;

            st        SendingIMUR(A2)       ; remember this!
            bsr       SendFrame             ; and send it
SndIMUR1    _SUBEND 'SENDIMUR'
            EJECT
;_______________________________________________________________________
;
;   ReadPacket - read in the specified number of bytes into the specified
;        buffer. It is an error to request more bytes than have been received.
```

(continued on page 88)

Breakthrough in interface management. Generate C code from Dan Bricklin's Demo screens. Date fields. Full color support. Money fields. Fully programmable field behavior. Scrolling text within fields. Calculator style numeric input. User definable entry validation. Field marking. Orthogonal field movement. Specify fields by number or location. Source code included. Screen sizes limited only by memory. Interfaces with db_VISTA and other libraries. Text style numeric input. Input masking. List fields. Create spreadsheets. Includes Look & Feel screen designer. Integer fields. String formatting commands. Date and time validation functions. Generate C code with Look & Feel screen designer. Supports automatic vertical and horizontal scrolling. Clean screen fields per screen limited only by development. String fields. Easy to painting. Bind as much data as de data entry with commas. Ask a gramming library. Hexadecimal or No fields. Float fields. Quick C. Speaker functions. Lattice. Create Slug. Numeric validation routines. keystroke level. Customize screens 30 day money back guarantee. Gen assortment of editing commands. windows. Assign validation data to credentials. Pull down menus. Sup mode. All functions are kept in C style function reference. Pop-up functions. Numeric range checking. tive function names. Date and time Capture screens from existing as deep as desired. Easy to main checking. Date and time conver definition language based on C's ly definable borders. The current cally highlighted. Create reports.

machine. Number of memory. Fast screen modify. Fast screen sired to fields. Numeric bout our linear pro fields. Long fields. Yes Read only fields. reports. Codename Validate data at the and menus at run time. eric data pointer. Rich Easy to learn. Pop-up fields. Corporate C ports EGA 43 line separate modules. Full prompt and message No royalties. Descrip- conversion routines. programs. Nest screens tain. Run time error sion functions. Screen printf. Time fields. Ful- field can be automati- Exploding borders.

C-scape 2.0

with

NEW!

Look&Feel™

The state-of-the-art interface management system preferred by professional C programmers and consultants worldwide.

Convert old programs to C. Borders with titles. Color map enables use of logical colors. Toll-free telephone support line. 24 hour bulletin board. Automatically detects type of monitor being used. ANSI driver included. Screen and field definitions. Uses device drivers for portability. View text in pop-up word entry fields. paging functions Customizable lect different cur Supports CGA, monochrome. cludes functions the display. In ANSI device dri ety of keyboard Lined borders. menuing systems. scroll lights. Vid ver included. drivers can be cre map enables log colors. Borders lines. Fully inte system. Create as as needed. Create screens. Easy to ual. Professional port. Includes

windows. Read only fields. Rich assortment of editing commands. Pass- Includes ROM BIOS driver. Fields can support any data type. Scrolling/ included. Specify writeable and non-writeable positions within fields. borders. Se sor types. EGA, and Aztec. In for writing to cludes an ver. A vari functions. Multi-level Borders with eo RAM dri New device ated. Color ical use of with prompt grated help many screens data entry follow man customer sup higher level

Look&Feel

- WYSIWYG screen design tool
- Generates readable C code
- Create menus and data entry screens
- Define fields of any type
- Variables, prompts, and validation
- Line draw and erase
- Block, move, cut, paste, copy
- Horizontal and vertical scrolling
- Edit Dan Bricklin Demo slides
- Full color support
- Fast, easy, and fun to use
- Includes help
- Full-feature demo available

C-scape 2.0

- Windows, windows, windows
- Menus, menus, menus
- Vast help system
- Create any type of field
- Data entry and validation
- Smart borders
- Extensive function library
- Swappable device drivers
- Easy to learn and use
- Easy to maintain and modify
- Unsurpassed flexibility
- Professional manual
- No royalties; no run-time license
- Source code included
- Demo package available

functions. Device drivers swappable at run-time. Context sensitive help system. Cross referenced help screens. Protected fields. Object-oriented design. Read in screen defini tions from disk files. Digitally mastered. Assign prompt strings to fields. UNIX. No run-time license. Numeric range checking. Unified field theory. Full printf % substitution within screen definitions. Supports all memory models. C Bricklin run. Turbo C. 24 hour bulletin board. Higher level functions included. Object-oriented design. All library functions are kept in separate modules. Nest screens as deep as desired. Design screens with Look & Feel screen designer. New device drivers can be created. Create as many screens as needed. No run-time license. Hexadecimal fields. Prefer-red by professionals and consultants. Microsoft. Cross referenced help system. by space aliens. Generate C code with Look & Feel screen designer. Context sensitive help system. Scroll lights. Read in screen definitions from disk files. Automatic vertical and horizontal scrolling. Batteries not included. Double and single line borders. Cross-referenced help system. Save and restore regions of the display. Nested menus. Quick C. Create screens from ASCII files. Easy to learn and use. Horizontal and vertical scrolling. Used by consultants and corporations worldwide. Easy to maintain. Professional documentation. Screen designer creates readable C code. Portable. Easily modifiable functions. No royalties. Source code included. Turn Dan Bricklin slides into C. Professional support. Interface examples for data base management. Validation at keystroke level. Vast integrated and indexed context-sensitive help system. Save and restore regions of the display. Now supporting Quick C, Turbo C, Aztec, Lattice, Microsoft, UNIX and others. And that's not all. Call for demo.

Oakland Group, Inc.

675 Massachusetts Avenue
Cambridge, MA 02139-3309

800-233-3733 CALL
617-491-7311 NOW

MasterCard VISA

PC/MS-DOS $279, plus shipping (includes C-scape, Look & Feel, source, manual and support). UNIX/others call. 30-day review.

CIRCLE 227 ON READER SERVICE CARD

Listing One *(Listing continued, text begins on page 18.)*

```
;
;   ReadRest - read in the rest of the packet, putting the specified number
;       of bytes into the specified buffer.  Error if packet longer than buffer.
;
;   Call:
;       A0,A1,A2 = SCC read and write addresses and local variables
;       A3 -> buffer to read into
;       A4 -> start of ReadPacket
;       D3 = byte count to read (word)
;
;   Return:
;       D0 changed
;       D1 number of chars still unread (ReadPacket); modified (ReadRest)
;       D2 saved
;       D3  = 0 if exact number of bytes requested were read
;           > 0 indicates number of bytes requested but not read
;               (packet smaller than requested maximum)
;           < 0 indicates number of extra bytes read but not returned
;               (packet larger than requested maximum)
;       A0,A1 preserved by ReadPacket, modified by ReadRest
;       A3 -> one past where last character went
;       A4,A5 saved (until packet's all in or error)
;
; NOTE: CRC bytes not included in counts
---------------------------------------------------------------------

ReadPacket  BRA.S    DoRP                  ; Need this for two entry points

ReadRest    movem.l  a0/a1/D2,-(sp)        ; save some regs
            move     RcvdLen(a2),D1        ; get the number of remaining chars in D1
            move     D1,D0                 ; we expect to copy D1 bytes
            move     #0,-(sp)              ; and expect good return status
            sub      D1,D3                 ; compute (D3 - D1)
            bpl.s    @1                    ; go if we should copy D1 bytes (it fits)
            add      D3,D0                 ; otherwise, copy D3 bytes (d1 + (d3-d1))
            move     #-1,(sp)              ; and set error return status
@1          movem.l  SaveA45(a2),a4/a5     ; restore A4 and A5
            bra.s    DoCopy                ; and go to the common code
```

PRODUCTION LANGUAGES CORP.

PRODUCTION QUALITY

68020

ANSI C Compiler

If you are doing serious development for the 68020 you already know that existing C compilers do not utilize the processor's full power . . . .

UNTIL NOW!

- Uses FULL 68020 instruction set
- Full 68881 floating point co-processor support
- Global optimization
- Full ANSI Libraries
- Cuncurrency supported

Available on a variety of hosts including VME, VAX, IBM PC & MAXII

CALL TODAY!

817-599-8366

Production Languages Corporation
P.O. Box 109, Weatherford, Texas 76086

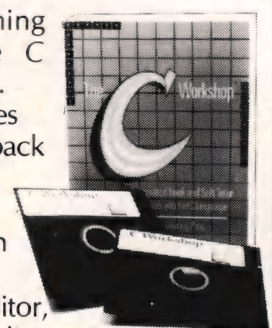
CIRCLE 399 ON READER SERVICE CARD

Easy to C

C is a great programming language. Now the C *WORKSHOP* makes it easy.

Interactive software teaches you C with immediate feedback on your program exercises.

The C *WORKSHOP* has everything you need to learn C and write your own programs, too. You get a fast editor, standard C compiler, and online help.



Let the other guy struggle with confusing books and compilers. Join AT&T and other major companies now using the *C WORKSHOP*. Columnist Adam Green calls it "the most intriguing new type of training system I've ever seen." (InfoWorld 1/27/86)

Order Information

To order the C *Workshop*, call toll-free (800) 227-2400 ext. 955 day or night (Visa, MC, or AmEx). Or send check to Wordcraft, 3827 Penniman Ave., Oakland, CA 94619. $69.95 plus $5.00 shipping (Priority Mail). In CA, add $4.90 sales tax.

Wordcraft
Quality software since 1981

```
DoRP      movem.l a0/a1/D2,-(sp)      ; push some regs
          move    RcvdLen(a2),D1      ; get the number of remaining chars in D1
          move    D1,D0               ; assume we'll copy them all
          move    #-1,-(sp)           ; and that there's an error
          sub     D3,D1               ; update D1 (remaining bytes in buf)
          bmi.s   DoCopy              ; go if it's negative (error)
          move    D3,D0               ; we'll read what they asked for (D3)
          clr     (sp)                ;  and remember that it's exactly right
          clr     D3                  ;

DoCopy    move.l  LAPStash(A2),a0     ; point at the source data
          ext.l   D0                  ; belt and suspenders (D0 = actual length)
          add.l   D0,LAPStash(A2)     ;    and update the LAPStash value
          sub     D0,RcvdLen(a2)      ;    and the num chars remaining
          move.l  A3,A1               ; point at the dest buffer
          lea     0(A3,D0),A3         ; update the return pointer
          _BlockMove                  ; Do It
          move    RcvdLen(a2),D1      ; return number of unread chars
          move    (sp)+,d0            ; get the return status back
          movem.l (sp)+,a0/a1/D2      ; get the other registers
          tst     D0                  ; set the CCR.
          rts
          EJECT

;______________________________________________________________________
;
; NextCRC -- compute a CRC on the word pointed at by A3 and the char in D0
;
;       This routine computes a CRC-16 on a stream of bytes.  It uses a
;       table lookup scheme to implement a x^16 + x^15 + x^2 + 1 polynomial.
;       The interested reader is referred to McNamara's Technical Aspects
;       of Data Communications, second edition, pps 110-122 for an obliquely
;       related discussion.
;
;       This routine takes the storage short cut of looking up two four-bit
;       values in a 16-entry table instead of one eight-bit value in a 256
;       word table.  This saves a considerable amount of space (32 bytes vs.
;       512 bytes for the table).
;
;       One pass thru this routine (one character) is about 262 cycles, or
;       33.45 usec on a Mac.  This is a data rate of ~29,900 char/sec,
;       or plenty fast to keep up with a 9600 baud link.
;
; Entry:  A3 -> CRC accumulator
;         D0    LSbyte is the data char (already masked to 8 bits)
;
; Exit:   D1,D2 changed
;         Other regs unchanged

;______________________________________________________________________

NextCRC _SUBR        0              ; for macsbug
        move         (a3),D2        ; D2 is the temp accumulator
        move         D0,D1          ; make a copy of the input character
; first work on the least significant nibble
        eor          D2,D1          ; xor the accumulator with the data char
        and          #$0F,D1        ;     to get an index into the CRCTable
        add          D1,D1          ; to make a word index
        lsr          #4,D2          ; shift the CRC right four bits
        move         CRCTable(D1),D1
        eor          D1,D2          ; and mask it with the approp. table entry
        move         D0,D1
        lsr          #4,D1          ; shift the data char right four bits
;  and do it again for the high nibble
        eor          D2,D1          ; xor the accumulator with the data char
        and          #$0F,D1        ;     to get an index into the CRCTable
        add          D1,D1          ; to make a word index
        lsr          #4,D2          ; shift the CRC right four bits
        move         CRCTable(D1),D1 ; and mask it with the approp. table entry
        eor          D1,D2
        move         D2,(a3)        ; remember this CRC for next time
        _SUBEND      'NEXTCRC '

CRCTable DC.W        $0000,$CC01,$D801,$1400
         DC.W        $F001,$3C00,$2800,$E401
         DC.W        $A001,$6C00,$7800,$B401
         DC.W        $5000,$9C01,$8801,$4400
            EJECT
;
. . .
```

(Listing One to be continued next month. Listings Two and Three will also appear next month.)

Announcing . . .

The

Programmer's Power Pack

Now you can reach 100,000 programmers, consultants, and systems integrators with your postcard ad. The *Programmer's Power Pack* card deck targets this elite audience, including subscribers to *Dr. Dobb's Journal of Software Tools*, for only a little over a penny a contact!

The Programmer's Power Pack Card Deck

Next Mailing Date: December 28, 1987
Reservation Closing: November 23, 1987

For advertising rates, card specifications, and to reserve your space, contact:

Ann Roskey
Northeast Account Manager
415-366-3600

Stephen Nestel
National Account Manager
415-521-4133

NEW! FASTER THAN EVER! DeSmet C V3.0

FASTER C DEVELOPMENT

Invoke the DeSmet C compiler from the SEE™ full screen editor and the first error will return you immediately to SEE at the error line with the error message displayed.

FASTER COMPILATION

When you don't use inline assembly code or don't want to see the ASM88 output, the V3.0 compiler produces object code directly - making DeSmet C up to ***twice as fast*** as before.

PLUS EXPANDED STANDARD LIBRARY

Networking, path, file, time, enhanced string functions, environment support now included.

FULL FEATURES WITH EVERY PACKAGE
— ONLY $109 —

C Compiler, Assembler, Binder, Librarian, Execution Profiler, Overlays, 8087 and S/W Floating Point, Full STDIO Library and Full Screen Editor (SEE). Debugger and Large Case options available at $50 each.

C Ware Corporation
P.O. Box 428, Paso Robles, CA 93447
Phone: (805) 239-4620 **Telex:** 358185 **BBS:** (805) 239-4834
We accept **VISA, MC & AMEX.** Call now and we'll ship today.
Street Address: 945 Spring #14, Paso Robles, CA 93446

Now with dBx EXPRESS

dBASE to C

the dBx™ Translator

- C from dBASE II, III, III+ programs
- Move to UNIX, XENIX, QNX, MAC, AMIGA
- Faster, more reliable IBM PC programs
- Supports multi-user and network
- Run your code on any standard C system
- Know dBASE? Learn C easily.
- Priced from $350; available from distributors
- Includes full screen handler library
- Supports a choice of C database managers

from Desktop Ai

1720 Post Road East, Westport, CT 06880
Telephone: 203-255-3400 • Telex: 6502972226MCI
MCIMAIL-DESKTOPAI

dBASE is a trademark of Ashton-Tate   dBx is a trademark of Desktop AI

CIRCLE 258 ON READER SERVICE CARD

FORTH PRELUDE

Listing One *(Text begins on page 40.)*

```
Listing 1

SCR# 0
( A Forth Standard Prelude )

This file defines additional functions and extensions which
cannot be provided in the Forth-83 Standard.

Select the appropriate prelude for the particular Forth you are
using and load it before loading a Standard program using these
functions.


Copyright 1986 1987 by Martin J. Tracy.

SCR# 1
( Select a FORTH-83 implementation)
FORTH DECIMAL

(     2 LOAD  ( *** model ****)
(   3 4 THRU  ( Lab Microsystems, Inc.  PC/FORTH  Ver. 3.0)
(     5 LOAD  ( Laxen-Perry                    F83  Ver. 2.1)
(     6 LOAD  ( MicroMotion MasterForth              Ver. 1.2)
(     7 LOAD  ( ZEN                                  Ver. 0.0)
(     8 LOAD  ( FORTH, Inc. PolyFORTH II     ISD-4  MS-DOS)

SCR# 2
( FORTH-83 functions-- typical definitions)
( Note: functions already provided need not be redefined.)
: RECURSE    [COMPILE] RECURSE ;  IMMEDIATE
: INTERPRET   INTERPRET ;

: I> ( - 'data )   COMPILE R> ;  IMMEDIATE
: >I ( - 'data )   COMPILE >R ;  IMMEDIATE

( Used for alignment: )
: ALIGN    HERE 1 AND ALLOT ;
: REALIGN  ( a - a' )   DUP  1 AND + ;

( Note: defined here for a dumb terminal.)
: TAB ( x y )   CR  2DROP ;
: PAGE   25 0 DO  CR  LOOP  0 0 TAB ;
: MARK ( a n)   TYPE ;

SCR# 3
( Lab Microsystems, Inc.  PC/FORTH  Ver. 3.0)

( RECURSE and INTERPRET are provided.)

( Used by hi-level run-time words to find in-line data: )
: I> ( - 'data )   COMPILE R> ;  IMMEDIATE
: >I ( - 'data )   COMPILE >R ;  IMMEDIATE

( PC/FORTH already has a word ALIGN which works differently.)
( The following two definitions must be in the order shown: )

: REALIGN  ( addr --- addr' )   ALIGN ;
( 8086/80286  run-time I realignment.)
: ALIGN  ( --)  EVEN ;
( 8086/80286  compilation address alignment.)


SCR# 4
( Lab Microsystems, Inc.  PC/FORTH  Ver. 3.0)

( x y --- )
: TAB   GOTOXY ;

( --- ; clear screen and home cursor )
: PAGE  CLS ;

( a n --- ; display string in inverse video )
: MARK   REVERSE TYPE  REVERSE ;



SCR# 5
( Laxen-Perry No Visible Support  F83  Ver. 2.1)
( RECURSE and INTERPRET are provided.)

( Used by hi-level run-time words to find in-line data: )
: I> ( - 'data )   COMPILE R> ;  IMMEDIATE
: >I ( - 'data )   COMPILE >R ;  IMMEDIATE

( Used for alignment: )
: ALIGN     ;    : REALIGN   ;

: PAGE  ( -- )         DARK ;
\ clear screen and home cursor.
: TAB    ( x y -- )   AT ;
\ move cursor to given coordinate.
: MARK  ( a n -- )    TYPE ;
\ the MARK function is not provided in F83.

SCR# 6
( MicroMotion MasterForth  Ver. 1.2.4)
( RECURSE is provided.)

: INTERPRET   TIB #TIB @ EVAL ;

: >I   COMPILE >R ;  IMMEDIATE
: I>   COMPILE R> ;  IMMEDIATE

NEED   ALIGN \IF  :   ALIGN ;
NEED REALIGN \IF  : REALIGN ;

( PAGE is provided.)
```

```
: TAB  ( x y)   AT ;
: MARK ( a n)  +INVERSE TYPE  -INVERSE ;


SCR# 7
( ZEN 0.0)

( RECURSE and INTERPRET are provided.)
( >I and I> are provided.)

NEED   ALIGN \IF  :   ALIGN ;
NEED REALIGN \IF  : REALIGN ;

( PAGE TAB and MARK are provided.)


SCR# 8
( FORTH, Inc. polyFORTH MS-DOS ISD)    HEX  1F1F WIDTH !
( Forth-83 Compatibility layer must be loaded first.)

: RECURSE   LAST @ @  2+  COUNT +  , ;  IMMEDIATE
( INTERPRET is provided.)

( Used by hi-level run-time words to find in-line data: )
: I> ( - 'data )   COMPILE R> ;  IMMEDIATE
: >I ( - 'data )   COMPILE >R ;  IMMEDIATE

( Used for alignment: )
: ALIGN     ;     : REALIGN    ;

( PAGE and MARK are provided.)
: TAB ( x y )    SWAP TAB ;
( move cursor to given coordinate.)
```

End Listing

Parallel Programming for "C"

INTERWORK

A Concurrent Programming Toolkit

Interwork is a "C" program library which allows you to write your programs as a set of cooperating concurrent tasks. Very useful for simulation, real-time applications, and experimentation with parallel programming.

FEATURES

- Supports a very large number of tasks (typically more than 100) limited only by available memory. Low overhead per task results in very fast context switching.
- Provides a full set of inter-task communication (ITC) facilities, including shared memory, locks, semaphores, blocking queues, and UNIX*-style signals. Also has building blocks for constructing your own ITC facilities.
- Handles interrupts (DOS version) and integrates them into task scheduling. Supply your own interrupt handlers or block tasks on interrupts.
- Lets you trace task switches and inter-task communication.
- Comes with complete documentation including a user's manual and reference manual of commands.

Interwork is available for the following systems:

Hardware	*Operating System*	*Price*
IBM PC, XT, AT	PC-DOS 2.0 or later	$129
IBM PC AT	XENIX*	$159
DEC VAX*, SUN III	UNIX 4.2BSD	$249

PC-DOS version is compatible with DeSmet, Lattice, and Microsoft C compilers.
Please specify hardware and operating system when ordering. Shipping and handling included; COD orders add $2.50. Send check or money order to:

Block Island Technologies
Innovative Computer Software
13563 NW Cornell Road, Suite 230, Portland, Oregon 97229-5892
(503) 241-8971

*Trademarks: UNIX, AT&T Bell Laboratories, Inc.; XENIX, Microsoft, Inc.; VAX, Digital Equipment Corporation

CIRCLE 263 ON READER SERVICE CARD

C Programmers: Combine C and COMMON LISP to Increase the Power of Your Software

TransLISP PLUS™

Simple.

Add LISP features to your software without making it a full time job. The TransLISP PLUS tutorial, on-line help, and 30 sample programs with commented source make it easy.

Practical.

Start by modifying the LISP sample programs and including them in a system you wrote in C. Yes, in C! TransLISP PLUS includes a C Language Interface that lets you integrate your Microsoft C code and libraries with all or portions of our LISP interpreter.

Use TransLISP PLUS to add natural or command language features to replace menus . . . or to flexibly manage related but disparate information. Code from C libraries provided by other vendors can be integrated into your program to perform tasks not normally part of LISP.

Thorough.

TransLISP PLUS took over 400 primitives from the most widely used and respected LISP standard, COMMON LISP, and made it available on IBM PCs, XTs, ATs, and virtually every other MSDOS machine. So now you can work with anything from a $700 PC to a $7000 PC.

The utilities toolbox is included at no charge with a built-in editor, pretty printer, cross reference, and additional debugging tools.

An optional Runtime encrypts your source code so that you can distribute your applications safely. You pay no royalties.

Requires MSDOS 2.0+, 320K RAM, and a 360K floppy.

MONEYBACK GUARANTEE

Try TransLISP PLUS ($195) for 30 days — if not satisfied get a full product refund. The Optional Runtime is available for $150. Or start by learning LISP with TransLISP ($95) then upgrade to PLUS for $158.

Call (800) 255-4659
In MA (617) 331-0800

The Coder's Source™ 541-D Main St., Suite 412, So. Weymouth, MA 02190

CIRCLE 315 ON READER SERVICE CARD

Listing One *(Text begins on page 46.)*

```
/*
 *
 *                              FINDCMD
 *
 *                           MODULE: MAIN.C
 *
 *                COPYRIGHT (C) 1987 by Charles F. Bowman
 *                        All Rights Reserved.
 *
 */

/*
 *      This module contains the driving loop of the program.
 */
#include        <stdio.h>

#include        "findcmd.h"

struct  attribs atts[] = {              /* For display */
        DIRECT, 'd',
        RDONLY, 'r',
        HIDDEN, 'h',
        SYSTEM, 's',
        ARCHIV, 'a'
};

char *month[] = {
        "Jan", "Feb", "Mar",
        "Apr", "May", "Jun",
        "Jul", "Aug", "Sep",
        "Oct", "Nov", "Dec"
};

main( ac, av )
int     ac;
char    *av[];
{
        int     i, len;
        char    dir[ MAXPATH ];
        char    path[ MAXPATH ];
        char    *tptr, tpath[ MAXPATH ];
        struct  dirent  dfile;

        if( ac == 1 ){
                fprintf( stderr, "usage: fc pat1 pat2 ... patn\n" );
                exit( 1 );
        }

        printf( "FINDCMD - COPYRIGHT (C) 1987, Charles F. Bowman. " );
        printf( "All Rights Reserved.\n\n" );

        sprintf( path, ".;%s", getenv("PATH") );        /* Get path */

        for( i = 1; i < ac; i++ ){
                /*
                 *      For each supplied pattern argument
                 */
                strcpy( tpath, path );
                tptr = tpath;
                inits( av[i] );         /* set-up transition table */
                while( (len = nextdir(tptr)) > 0 ){
                        /*
                         *      For each directory component
                         */
                        tptr[ len ] = NIL;
                        if( tptr[len-1] == '\\' ){
                                /*
                                 *      No double backslashes!
                                 */
                                sprintf( dir, "%s*.*", tptr );
```

```
                        } else {
                                sprintf( dir, "%s\\*.*", tptr );
                        }

                        /*
                         *      Compare each entry in directory
                         */
                        if( firstf(dir, &dfile) ){      /* Get first */
                                /*
                                 *      Error - bad dir in path
                                 */
                                fprintf( stderr,
                                        "BAD DIRECTORY SEGMENT: [%s]\n",
                                        dir );
                                break;
                        }
                        do{
                                if( state(slcase(dfile.dname)) ){
                                        /*
                                         *      Print if a match is found!
                                         */
                                        putfile( &dfile, tptr );
                                }
                        } while( nextf(&dfile) );       /* Get next */

                        tptr += len + 1;
                }
        }

        exit( 0 );
}

/*
 *      NEXTDIR: return the next directory component of 'PATH'
 */
nextdir( cp )
char    *cp;
{
        register        count;


        count = 0;
        if( *cp == NIL ){                       /* End of list */
                return( -1 );
        }
        while( *cp != NIL && *cp != FLDSEP ){
                cp++;
                count++;
        }
        return( count );
}

/*
 *      SLCASE: convert a string to lower case
 */
char    *
slcase( cp )
char    *cp;
{
        register char   *ptr;

        ptr = cp;
        while( *ptr ){
                *ptr = tolower( *ptr );
                ptr++;
        }

        return( cp );
}

/*
 *      PUTFILE: display directory info for each matched file
```

(continued on next page)

MetaWINDOW

Power Graphics for your PC!

PC TECH JOURNAL

"Product of the Month"
"... a technological tour de force for fast PC graphics."

NO ROYALTIES!

MetaWINDOW is an advanced, high performance graphics development toolkit which bridges the gap between low-level graphic primitive libraries and pre-packaged window managers.

Unparalleled Performance!

MetaWINDOW provides an expanded set of graphic drawing functions, plus the added functionality and performance required for designing multi-window desktop applications.

- auto-cursor tracking
- pull-down menus
- pop-up windows
- comprehensive graphic functions
- multiple fonts

Enhanced Features!

- Display multiple bitmap or "filled-outline" fonts.
- Face fonts for bold, italic, underline or strike-out stylings.
- Full "RasterOp" transfer functions for writing, erasing, rubberbanding or dragging: lines, text, icons, bit images and complex objects.
- Create pop-up menus, windows and icons.
- Supports IBM's new PS/2 VGA and MCGA graphics.

MetaWINDOW comes complete with langauge bindings for 16 popular C, Pascal and Fortran compilers, plus dynamic runtime support for over 40 graphics adaptors and input devices.

MetaWINDOW
Advanced Graphics Toolkit
*4 disks, 3 260 page manuals - $195**

NEW! - TurboWINDOW/C
*All the features of MetaWINDOW for Borland Turbo C! - $ 95**
** Plus $5.00 shipping and handling*

TO ORDER CALL 1-800-332-1550
For information or in CA call 408-438-1550

METAGRAPHICS
SOFTWARE CORPORATION
269 Mount Hermon Road
Scotts Valley, CA 95066

CIRCLE 392 ON READER SERVICE CARD

Hard Locks for Soft Parts.

At Rainbow Technologies, we think protecting software developers' investments is very serious business. That's why we designed the first fully effective security solution for software running on PCs and other computers.

Our family of virtually impenetrable Software Sentinel hardware keys provides the highest level of software protection the developer can get. While remaining invisible to the end user.

Take a look.

Key Sentinel Family Features.

Prohibits unauthorized use of software ▫ No need for copy protection ▫ Unlimited backup copies ▫ Virtually unbreakable ▫ Pocketsize key ▫ Transparent operation ▫ Transportable

Software Sentinel.

- Runs under DOS and Xenix, on IBM PC/XT/AT and compatibles
- Algorithm technique (Never a fixed response)
- Serial or parallel port version
- Minimal implementation effort
- Higher level language interfaces included
- 100 times faster than fixed-response devices (1ms)

Software Sentinel-C.

- For developers who want to customize or protect multiple packages with one device
- 126 bytes of non-volatile memory that is programmed before shipment of software
- We supply a unique programming adapter for programming the unit
- Higher level language interfaces included
- Runs under DOS on PC/XT/AT and compatibles
- Parallel port version only

Software Sentinel-W.

- Designed for workstations, supermicros and minicomputers
- Serial port only (modem-type)
- Algorithm technique
- We provide detailed interface specifications: Developer creates a port driver
- Interface requirements: 25 pin DB25P or DB25S; RS232/RS422/RS423
- Only signals used: DTR & RTS from computer; signal ground; DSR or optional DCD from Software Sentinel-W or external device. TXD, RXD, CTS, RI passed through.

Call For Software Sentinel Evaluation Kit Pricing.

International Distributors Wanted

RAINBOW TECHNOLOGIES
18011-A MITCHELL SOUTH IRVINE, CA 92714 USA
(714) 261-0228 TELEX: 386078 FAX: (714) 261-0260

CIRCLE 255 ON READER SERVICE CARD

Listing One

(Listing continued, text begins on page 46.)

```
 */
putfile( dfile, dir )
struct  dirent  *dfile;
char    *dir;
{
        int     i;
        char    datestr[ 50 ];

        /*
         *      Print attribute chatacters
         */
        for( i = 0; i < SA_SIZE(atts); i++ ){
                if( atts[i].val & dfile->dattr ){
                        putchar( atts[i].chr );
                } else {
                        putchar( '-' );
                }
        }

        fdosdte( datestr, &dfile->ddate );
                        /* DOS -> ASCII */
        if( dir[strlen(dir)-1] == '\\' ){
                /*
                 *      No double backslashes!
                 */

                printf( " %8D %s %s%s\n",
                        dfile->dsize,
                        datestr,
                        slcase(dir),
                        dfile->dname
                );
        } else {
                printf( " %8D %s %s\\%s\n",
                        dfile->dsize,
                        datestr,
                        slcase(dir),
                        dfile->dname
                );
        }

        return( 0 );
}

/*
 *      FDOSDATE: convert a DOS format date to an
                                        ASCII string
 */
fdosdte( where, dptr )
char    *where;
struct  dosdate *dptr;
{
        sprintf( where, "%s %2d %02d:%02d:%02d %4d",
                month[ dptr->month-1 ],
                dptr->day,
                dptr->hour,
                dptr->min,
                dptr->sec * 2,
                dptr->year + 1980
        );

        return( 0 );
}
```

End Listing One

Listing Two

```
/*
 *
 *                              FINDCMD
 *
 *                          MODULE: DOS.C
```

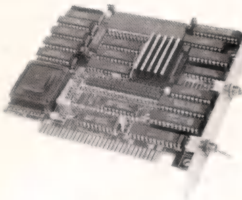
(continued on page 96)

MICROWAY ACCELERATES YOUR PC!

FastCACHE-286™

Runs your PC Faster than an AT!

Runs the 80286 at 9 or 12 MHz and the 80287 at 8, 9 or 12 MHz. Includes 8 kbytes of 55ns CACHE.



Compatible with IBM PC, XT, Leading Edge Model D, Compaq, and Turbo motherboards. Includes 8088 Reboot Switch, DCache, Print Spooler and Diagnostics .... **From $399**

LOTUS/INTEL EMS SPECIFICATION BOARDS

MegaPage™ The only EMS board which comes populated with two megabytes of cool-running, low power drain CMOS RAM installed. Includes RAM disk, print spooler, disk cache and EMS drivers. For the IBM PC, XT and compatibles... **$549**

MegaPage with ØK **$149**

MegaPage with 2 megabytes of HMOS RAM **$419**

MegaPage AT/ECC™ EMS card for the PC AT and compatibles includes Error Correction Circuitry. With ECC, 11 RAM chips cover 256K so the user never encounters RAM errors. With 1 megabyte CMOS RAM **$699**

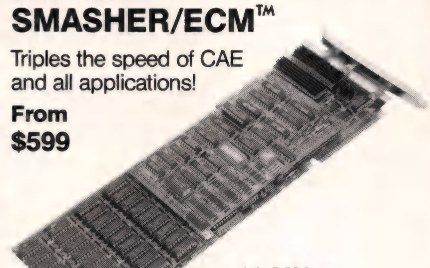
INTEL, JRAM, or Maynard .... CALL

INTEL INBOARD 386 ØK $1250

NUMBER SMASHER/ECM™

Triples the speed of CAE and all applications!

From $599



12 MHz 8086/8087 Accelerator Plus A Megabyte for DOS!

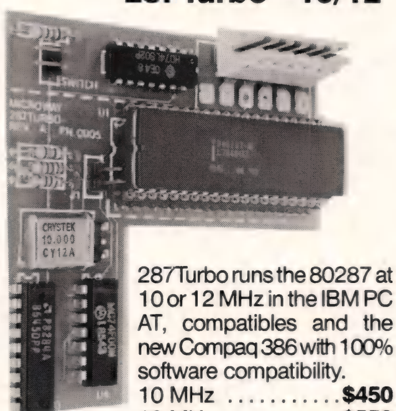
For the IBM PC, XT and compatibles

PC Magazine **"Editor's Choice"**

8087 SOFTWARE

IBM BASIC COMPILER	**$465**
MICROSOFT QUICK BASIC	**$79**
87BASIC COMPILER PATCH	**$150**
87BASIC/INLINE	**$200**
IBM MACRO ASSEMBLER	**$155**
MS MACRO ASSEMBLER	**$99**
87MACRO/DEBUG	**$199**
MICROSOFT FORTRAN V4	**$299**
RM FORTRAN	**$399**
LAHEY FORTRAN F77L	**$477**
MS or LATTICE C	**CALL**
STSC APL★PLUS/PC	**$450**
STSC STATGRAPHICS	**$675**
SPSS/PC+	**$695**
87SFL Scientific Functions	**$250**
87FFT	**$200**
OBJ → ASM	**$200**
PHOENIX PRODUCTS	**CALL**

287 Turbo™-10/12



287Turbo runs the 80287 at 10 or 12 MHz in the IBM PC AT, compatibles and the new Compaq 386 with 100% software compatibility.

10 MHz **$450**

12 MHz **$550**

PC Magazine **"Editor's Choice"**

8087 UPGRADES

All MicroWay 8087s include a one year warranty, complete MicroWay Test Program and installation instructions.

8087 5 MHz.................... **$99**
For the IBM PC, XT and compatibles

8087-2 8 MHz **$154**
For Wang, AT&T, DeskPro, NEC, Leading Edge

80287-3 5 MHz............... **$159**

80287-6 6 MHz............... **$179**
For 8 MHz AT and compatibles

80287-8 8 MHz............... **$259**
For the 8 MHz 80286 accelerator cards

80287-10 10 MHz............ **$395**

80387-16 16 MHz............ **$495**

PC-PAL™ Programmer...... **$395**

64K 150ns .....$15   256K 150ns $36

Call for great prices on V20 & V30

AT8™

Turns your AT into a high speed, multi-user Xenix business system!

8 port, intelligent serial controller with 3% response degradation. Includes 8 MHz 80186 with built in DMA **$1299**

MICROWAY SOFTWARE FOR LOTUS 1-2-3™

PowerDialer® Add-In for Lotus 1-2-3 Release 2. Automated telephone dialing from within 1-2-3. Adds least cost routing, automatic carrier selection and automated phone book worksheet. Builds customized dialing applications. Can be used with DesqView.................. **$79**

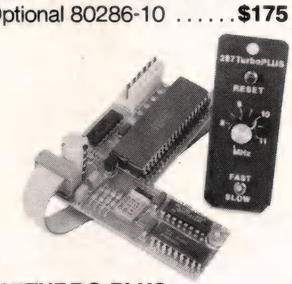
FASTBREAK™ employs the 8087 to increase the speed of Lotus 1-2-3™ Version 1A or 1A*. Users are reporting speed ups of between 3 and 36 to 1. When run with our NUMBER SMASHER accelerator card, recalculation speed ups of 10 to 30 are being reported **$79**

HOTLINK™ adds easy linking of spreadsheets to Lotus 1-2-3 Version 1A... **$99**

287 TURBO-PLUS™

Speeds up your AT

Adjustable 80286 Clock 6-12 MHz
10 MHz 80287 Clock
Plus Full Hardware Reset **$149**
Optional 80286-10 ...... **$175**



287TURBO-PLUS

With 80287 10 MHz **$549**

With 80287 12 MHz **$629**

CALL (617) 746-7341 FOR OUR COMPLETE CATALOG

MicroWay P.O. Box 79 Kingston, Mass. 02364 USA (617) 746-7341

The World Leader in 8087 Support!

MicroWay Europe
32 High Street
Kingston-Upon-Thames
Surrey England KT1 1HL
Telephone: 01-541-5466

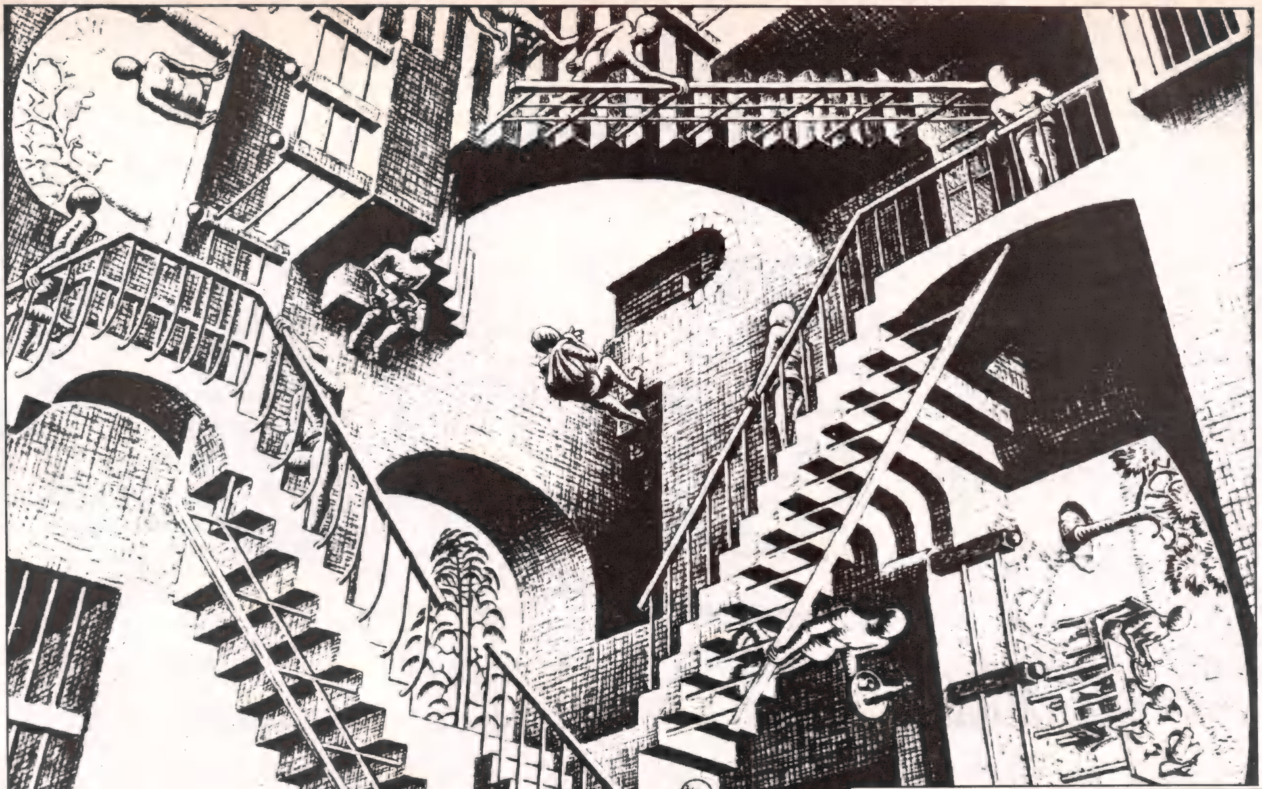
CIRCLE 300 ON READER SERVICE CARD

Listing Two *(Listing continued, text begins on page 46.)*

```
 *
 *              COPYRIGHT (C) 1987 by Charles F. Bowman
 *                      All Rights Reserved.
 *
 */

/*
 *      This module contains the DOS dependent functions to
 *      access file names in directories. This is non-portable code.
 */
#include        <dos.h>
#include        <stdio.h>

#include        "findcmd.h"

struct  reg     regs;                   /* set & retrieve regs */
static  struct  dirent  lfile;          /* DOS disk trans addr */

/*
 *      FIRSTF: initiate DOS environment and return first file name
 */
firstf( dirpath, dfile )
char    *dirpath;
struct  dirent  *dfile;
{
        /*
         *      Set disk transfer address
         */
        regs.r_ax = SETDTA;
        ptoreg( dsreg, regs.r_dx, regs.r_ds, &lfile );
        intcall( &regs, &regs, DOSINT );

        /*
         *      Find first
         */
        regs.r_ax = NFFIRST;
        regs.r_cx = HIDDEN | SYSTEM | DIRECT | RDONLY | ARCHIV; /* All! */
        ptoreg( dsreg, regs.r_dx, regs.r_ds, dirpath );
        intcall( &regs, &regs, DOSINT );

        if( regs.r_flags & F_CF ){
                /*
                 *      Error!
                 */
                return( 1 );
        }

        *dfile = lfile;
        return( 0 );
}

/*
 *      NEXTF: return all subsequent files in directory
 */
nextf( dfile )
struct  dirent  *dfile;
{
        /*
         *      Call DOS: find next
         */
        regs.r_ax = NFNEXT;
        regs.r_cx = HIDDEN | SYSTEM | DIRECT | RDONLY | ARCHIV;
        intcall( &regs, &regs, DOSINT );
        if( regs.r_flags & F_CF ){
                /*
                 *      Error!
                 */
                return( 0 );
        }

        *dfile = lfile;
        return( 1 );
}
```

End Listing Two

(Listing Three begins on page 106.)



Avoid extra steps.

You've got better things to do than repeat the same steps. Over . . . and over . . . and over. Up your productivity with Greenleaf Software.

With more than 70 new functions added to our popular libraries, Greenleaf is now the most complete and mature C language function resource available. It's no wonder we've been rated the best. Winning program developers in major corporations such as IBM, EDS and GM have proven our reliability in thousands of applications.

Step Lively

New Greenleaf Functions v.3.10 includes 295 of the functions you've been asking for — DOS, disk, video, color text and graphics, string, time/date, keyboard, plus many more! With Greenleaf, you'll finish faster.

Cut Corners

When it comes to merging information, the new Greenleaf Comm Library v.2.10 is the fastest communications facility of its kind. Over 120 functions — ring buffered, interrupt-driven asynchronous communications. And, only Greenleaf gives you the power to build a 16-port communication system.

Get on the Fast Track

Order your new Greenleaf library today! See your dealer or call 1-800-523-9830.

Greenleaf Comm Library	$185.00
Greenleaf Functions	$185.00
Greenleaf DataWindows	$225.00
Greenleaf C Sampler	$ 94.50
Digiboard Comm4	$325.00
Digiboard Comm8	$535.00

In stock, shipped next day.

Greenleaf DataWindows and Turbo C

DataWindows, the finest C programming windows tool available, puts windows, transaction data entry and menus at your fingertips.

Our new TURBO C versions are ready to get you going fast! And, our new 3-in-1 C Sampler for only $94.50 supports both Turbo C and Quick C with comm, windows, menus and more! Our libraries support all popular C compilers for MS DOS.

GREENLEAF Software

Call Toll Free:

800-523-9830

In Texas and Alaska:

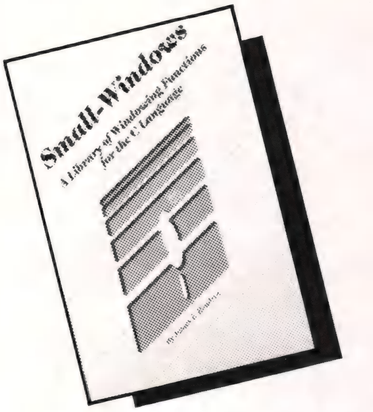
214-248-2561

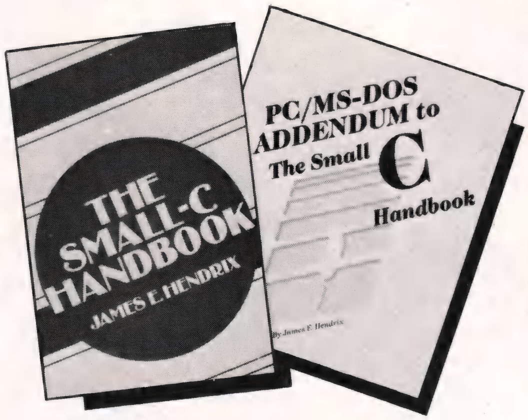
Greenleaf Software, Inc.
16475 Dallas Parkway, Suite 570
Dallas, Texas 75248

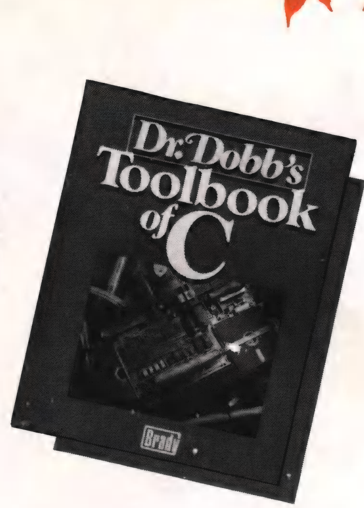
CIRCLE 97 ON READER SERVICE CARD

M&T BOOKS AND SOFTWARE TOOLS

C Toolbox







Small-Windows: A Windowing Library

Small-Windows is a complete windowing library for C and Small-C. The package contains: 18 video functions written in assembly language; 7 menu functions that support both static and pop-up menus; and 41 window functions, including functions to clean, frame, move, hide, show, scroll, push, and pop windows. A file directory facility illustrates the use of the window menu functions and provides file selection, renaming and deletion capability. Two test programs are also included. For PC/MS-DOS systems, and Microsoft C Version 4.0 and Small-C compilers. Documentation and full source code is included.

► Small-Windows Item #109 $29.95

Small-Tools: Programs for Text Processing

These Small-C programs perform specific, modular operations on text files, including: editing, formatting, sorting, merging, listing, printing, searching, changing, transliterating, copying, concatenating, encrypting and decrypting, and more. Supplied as source code. With the *Small-C Compiler* you can select and adapt these tools to meet your own needs. Documentation is included.

► Small-Tools Item #010A $29.95

Small-C Compiler & Small-C Handbook

Like a home-study course in compiler design, *the Small-C Compiler* and the *Small-C Handbook* provide everything you need for learning how compilers are constructed, and for learning C at its most fundamental level. Full source code is included.

► CP/M Small-C Compiler with Handbook
Item #006B $37.90

MS/PC-DOS Small-C Compiler with Handbook and Addendum
Item #006C $42.90

Small-Mac: An Assembler for Small-C

This assembler includes: a simplified macro facility, C language expression operators, object file visibility, descriptive error messages, and an externally defined instruction table. You'll receive the macro assembler, linkage editor, load-and-go loader, library manager, CPU configuration utility, and a utility to dump relocatable files. Documentation is included. For CP/M systems only.

Small-Mac Item #012A $29.95

► C Disk Formats
Please indicate MS/PC-DOS or CP/M. For CP/M specify: Apple, Osborne, Kaypro, Zenith Z-100 DS/DD, 8" SS/SD.

Dr. Dobb's Toolbook of C

This authoritative reference contains over 700 pages of the best C articles and source code from *Dr. Dobb's Journal*, along with new material by C experts. The level is sophisticated and pragmatic, appropriate for professional C programmers. You'll find hundreds of pages of valuable C source code, including a complete compiler, an assembler, and text processing utilities.

► Toolbook of C Item #005 $29.95

Special Packages

CP/M C Package

Save over $27!

Receive: Dr. Dobb's Toolbook of C, The Small-C Handbook and Small-C Compiler, Small-Mac Assembler, and Small-Tools Text Processing Programs. Only $99.95!

► CP/M Package Item #005A $99.95

MS/PC-DOS C Packages

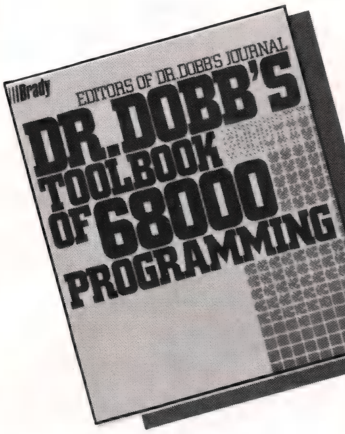
Save $22!

Receive: Dr. Dobb's Toolbook of C, The Small-C Handbook and MS/PC-DOS Addendum, Small-C Compiler, Small-Tools Text Processing Programs and Small Windows. Only $109.95! Specify Microsoft C Version 4.0 or Small-C compiler.

MS/PC-DOS Package
Item #005W $109.95

M&T BOOKS AND SOFTWARE TOOLS

Assembly Language Programming for the 68000 & Z80



Dr. Dobb's Toolbook of 68000 Programming

This complete collection of practical programming tips and techniques for the 68000 family includes the best articles on 68000 programming ever published in Dr. Dobb's, along with much new material.
Contents include:

An Introduction to the 68000 Family
- 68000 Instruction Set

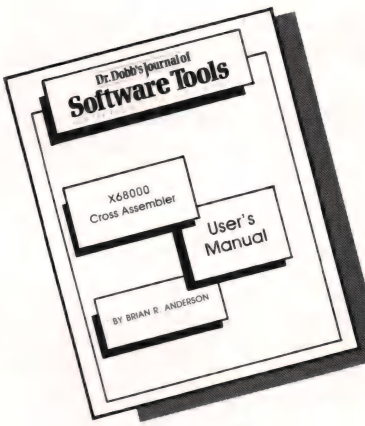
Development Tools
- Bringing Up the 68000: A First Step
- A 68000 Cross-Assembler

Useful 68000 Routines and Techniques
- A Simple Multitasking Kernel for Real-Time Applications
- The Worm Memory Test
- A Mandelbrot Program for the Macintosh

All programs are also available on disk!

► **68000 Toolbook**	**Item #040**	**$29.95**
68000 Toolbook with disk	**Item #041**	**$49.95**

Specify MS-DOS, CP/M, CP/M 8," Osborne, Macintosh, Amiga, Atari 520st.

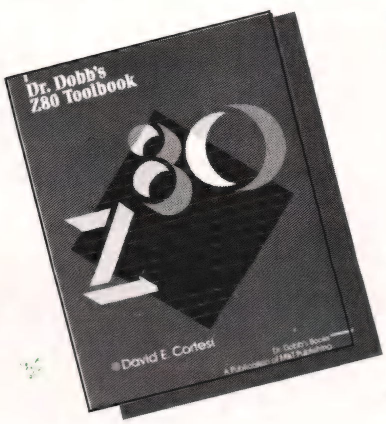


68000 Cross Assembler

An executable version of the 68000 Cross-Assembler discussed in the book is also available, complete with source code and documentation. Requires CP/M 2.2 with 64k or MS-DOS with 128k.

► **68000 Cross Assembler**	**Item #042**	**$25.00**

Specify 8" SS/SD, Osborne, or MS-DOS.



Dr. Dobb's Z80 Toolbook

by David E. Cortesi

Dr. Dobb's Z80 Toolbook puts the power of assembly language in the hands of anyone who's done a little programming. You'll find:

- A method of designing programs and coding them in assembly language and a demonstration of the method in the construction of several complete, useful programs.
- A complete, integrated toolkit of subroutines for arithmetic, string-handling, and total control of the CP/M file system.
- Every line of the toolkit's source code is there for you to read.

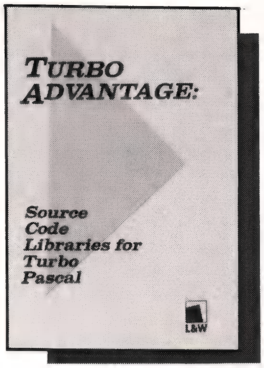
All the software—the programs plus the entire toolkit, both as source code and object modules for both CP/M 2.2 and CP/M Plus—is yours on disk. Most of the programs are included in the book, however, the disk is necessary for complete listings. A DRI RMAC assembler or equivalent is required.

► **Dr. Dobb's Z80 Toolbook**	**Item #022**	**$25**
Dr. Dobb's Z80 Toolbook with disk	**Item #022A**	**$40**

Specify 8" SS/SD, Apple, Osborne or Kaypro.

M&T BOOKS AND SOFTWARE TOOLS

Turbo Pascal Tools



TURBO Advantage:

Source Code Library for Turbo Pascal

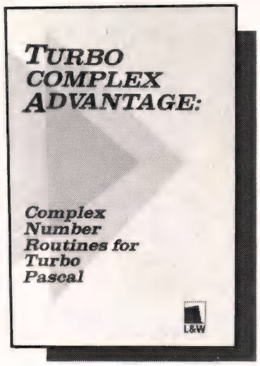
This library of more than 220 routines—complete with source code, sample programs and documentation—will save you hours of work developing and optimizing your programs!

Routines are organized and documented under the following categories: bit manipulation, file management, MS-DOS support, sorting, string operations, arithmetic calculations, data compression, differential equations, Fourier analysis and synthesis, matrices and vectors, statistics, and much more! All source code is included.

A detailed manual includes a description of the routine, an explanation of the methods used, the calling sequence, and a simple example. For MS/PC-DOS systems.

TURBO Advantage: Source Code Libraries for Turbo Pascal* is also available with *TURBO Advantage Complex: Complex Number Routines for Turbo Pascal* and *TURBO Advantage Display: Form Generator for Turbo Pascal.

► Turbo Advantage Item #070 $49.95



TURBO Advantage Complex:

Complex Number Routines for Turbo Pascal

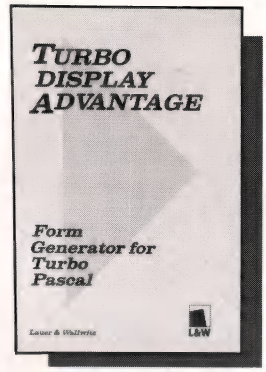
Working with complex numbers is easy with the Turbo Pascal procedures and routines provided in TURBO Advantage Complex!

TURBO Complex provides procedures for performing all the arithmetic operations and necessary real functions with complex numbers. Each procedure is based on predefined constants and types. By using these declarations the size of arrays are easily adapted. Each type declaration is a record with both a real and imaginary part. Use these procedures to build more sophisticated functions in your own programs.

TURBO Complex also demonstrates the usage of these procedures in routines for vector and matrix calculation with complex numbers and variables; simultaneous Fourier transforms; calculations of convolution and correlation functions; low-pass, high-pass, band- pass and band-rejection digital filters; and in solving linear boundary-value problems.

Source code and documentation is included for MS-DOS systems. Some of the *TURBO Complex* routines are most effectively used with routines contained in *TURBO Advantage.*

► TURBO Advantage/
Complex Package Item #070A $115
TURBO Complex Item #071 $89.95



TURBO Advantage Display:

Form Generator for Turbo Pascal

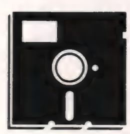
Now, even if you have little programming knowledge, you can design and process forms to fit your needs!

TURBO Display includes a menu-driven form processor, 30 Turbo Pascal procedures and functions to facilitate linking created forms to your program, and full source code and documentation. For MS-DOS systems.

Some of the *TURBO Advantage: Source Code Libraries for Turbo Pascal* routines are necessary to compile *TURBO Display.* You save $20 when you order *TURBO Advantage: Source Code Libraries for Turbo Pascal* together with *TURBO Display: Form Generator for Turbo Pascal.* Receive both for only $99.95!

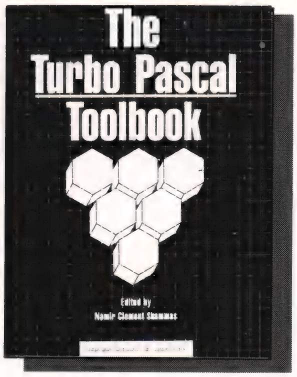
***TURBO Display: Form Generator for Turbo Pascal* is also available individually for $69.95.**

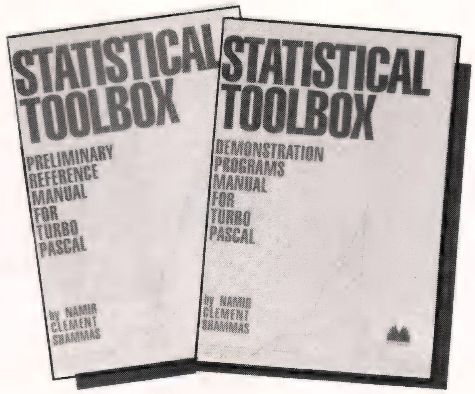
► TURBO Advantage/
Display Package Item #070B $99.95
TURBO Display Item #072 $69.95

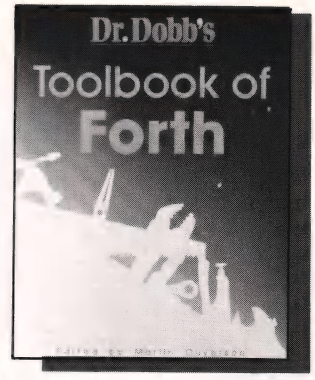


Each Turbo Advantage package includes complete source code on disk.

Programming Eloquence







The Turbo Pascal Toolbook

Edited by Namir Clement Shammas

Make your programming easier and more powerful with *The Turbo Pascal Toolbook!*

You'll find:

- an extensive library of low-level routines
- external sorting and searching tools, presenting a new database routine that combines the best features of the B-tree, B+ and B++ trees
- window management, to help you create, sort, and overlay windows
- artificial intelligence techniques
- mathematical expression parsers, offering two routines that convert mathematical expressions into RPN tokens
- a smart statistical regression model that searches for the best regression model to represent a given set of data.

All routine libraries and sample programs are on disk for MS-DOS systems, and over 800K of Turbo Pascal source code is included!

Turbo Pascal Toolbook
Item #080 $25.95
Turbo Pascal Toolbook with disk
Item #081 $45.95

STAT Toolbox

for Turbo Pascal

Bring convenience, power, and versatility to your statistics programs!

Two statistical packages in one!

A library disk and reference manual
Use these powerful statistical routines to build your applications.
Routines include:

- statistical distribution functions
- random-number generation
- basic descriptive statistics
- parametric and non-parametric statistical testing
- bivariate linear regression, multiple and polynomial regression.

A demonstration disk and manual
This package incorporates the library of routines into a fully functioning statistical program. Two data management programs are included to facilitate the storage and maintenance of data.

Full source code is included. (For IBM PC's and compatibles. Turbo Pascal version 2.0 or later, and PC DOS 2.0 or later are required).

STAT Toolbox **Item #050** $69.95

In Calif. 800-356-2002

Dr. Dobb's Toolbook of Forth

This comprehensive collection of useful Forth programs and tutorials contains *DDJ's* best Forth articles, expanded and revised along with new material. You'll find sections on:

- Mathematics in Forth
- Modifications/Extensions
- Forth Programs
- Forth—the Language
- Implementing Forth

You'll also find Appendixes that will help you convert fig Forth to Forth-83, and tell you how to stay up-to-date on the latest developments and refinements of this popular language.

The screens in the book are also available on disk as ASCII files.

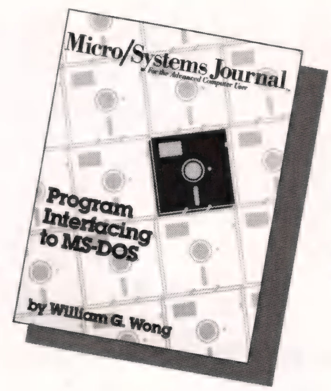
Dr. Dobb's Toolbook of Forth
Item #030 $22.95

Dr. Dobb's Toolbook of Forth with Disk
Item #031 $39.95

Please specify MS/PC-DOS, Apple II, Macintosh, or CP/M. For CP/M disks, specify Osborne or 8" SS/SD.

M&T BOOKS AND SOFTWARE TOOLS

Programming & Productivity Tools



Taming MS-DOS
by Thom Hogan

Taming MS-DOS takes you beyond the basics, picking up where your MS-DOS manual leaves off. You'll learn how to maximize your batch files with routines using redirection, filters and pipes, and routines that prevent accidental reformatting of your hard disk, redefine function keys, and locate files within subdirectories. Other batch files will implement an MS-DOS help system, including help text files, a menu system that interprets keyboard input, and a routine for quick redefinition of function keys.

You'll also learn how to create configurable AUTOEXEC.BAT files, and how to customize CONFIG.SYS and use ANSI.SYS to change the appearance of MS-DOS. **Taming MS-DOS** includes ready-to-use assembly language programs that enhance MS-DOS. You can rename directories and disk volumes, change file attributes, check available RAM and disk memory, display a memory-resident clock, and assign MS-DOS commands to ALT keys.

The programs, including batch files and MS-DOS enhancements, are available on disk with full source code.

► Book & Disk (MS-DOS)
$34.95 Item #59-3
► Book
$19.95 Item #24-0

Program Interfacing to MS-DOS
by William G. Wong

Originally featured in *Micro/Systems Journal,* this reprint provides ten concise chapters that will orient any experienced programmer to the MS-DOS environment. Topics include: program construction, character base input and output functions, and file access, including an in-depth discussion of CP/M style vs. Unix-style MS-DOS file access.

Program Interfacing to MS-DOS also contains sample program files, and a detailed description of how to build device drivers. The complete ten-part **Program Interfacing to MS-DOS** manual is available, along with the device driver for a memory disk and a character device driver, on disk with macro assembly source code.

► Manual & Disk (MS-DOS)
$29.95 Item #34-8

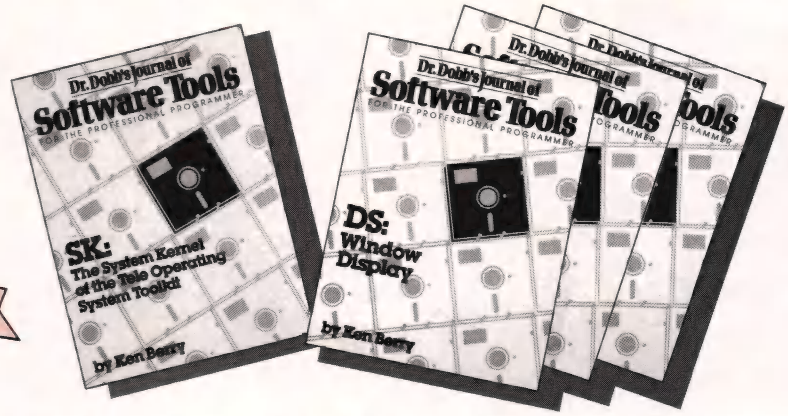
The New BASICs: Programming Techniques and Libraries
by Namir Clement Shammas

This book will orient the advanced programmer to the syntax and programming features of the new BASICs, including Turbo BASIC, Quick BASIC, and True BASIC. You'll learn the details of implementing subroutines, functions, and libraries to permit more structured coding. **The New BASICs** contains many programming examples and ready-to-use libraries, including libraries for string manipulation, extended string management, and sorting and searching. You'll also find a binary tree library, MS-DOS files library, a library for Pascal-like sets, and much more. Programs and subroutines are also available on disk, with full source code. MS-DOS format. Available in September 1987.

► Book & Disk (MS-DOS)
$39.95 Item #43-7
► Book
$24.95 Item #37-2

In Calif. 800-356-2002

Multitasking Operating System



The Tele Operating System Toolkit

by Ken Berry

The unique features of this four part, multitasking operating system will allow you to fully exploit the power of any 8086-based machine! *Tele* is written in C and assembly language for IBM PC compatibles and includes **preemptive multitasking capabilities and an unlimited number of tasks.** *Tele* contains **full C and Assembler source code,** as well as precompiled libraries. It is compatible with MS-DOS, Unix, and the MOSI standard. MS-DOS disk format.

SK: The System Kernel

SK includes the most crucial part of the Tele Operating System—the preemptive multitasking algorithm. *SK* also contains an initialization module, general purpose utility functions for string and character handling, format conversion, terminal support, and machine interface, along with a real-time task management system. All other components require *SK: The System Kernel.*

► Manual & Disk (MS-DOS)
$49.95 Item #30-5

DS: Window Display

DS contains BIOS level drivers for a memory-mapped display (the fastest way to display data), window management support and communication coordination between the operator and tasks in a multitasking environment. *DS* includes funtions to create and delete virtual displays, and functions to overlay a portion of a virtual display on the physical display. An unlimited number of virtual displays can belong to any particular task, and an unlimted number can be in the system at any time. Requires *SK: The System Kernel.*

► Manual & Disk (MS-DOS)
$39.95 Item #32-1

FS: The File System

FS supports MS-DOS disk file structures and serial communications channels. *FS* manages the storage of information on disks with a Unix-like file allocation method, and is compatible with both Unix and MS-DOS. *FS* also features a telecommunications support facility that allows a common set of functions to handle both disk files and telecommunications. Requires *SK: The System Kernel.*

► Manual & Disk (MS-DOS)
$39.95 Item #65-8

XS: The Index System

XS implements a tree-structured free-form data base. *XS* allows names and data of variable length with no practical limitation on data size. The algorithm used optimizes access for different processors and disk speeds. Requires *SK: The System Kernel and FS: The File System.*

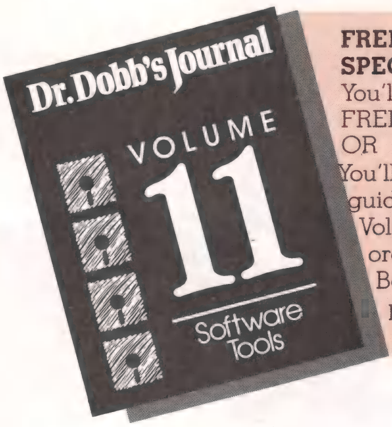
► Manual & Disk (MS-DOS)
$39.95 Item #66-6

The Tele Operating System Demonstration Disk

Give the Tele Operating System a try for only $5! This demo disk includes a working sample of the Tele Operating System. MS-DOS disk format.

► Demo Disk (MS-DOS)
$5.00 Item #70-4

Dr. Dobb's Bound Volumes



FREE!
SPECIAL FALL OFFER!
You'll receive all the listings from 1986 on disk FREE when you order Bound Volume 11
OR
You'll receive a FREE index on disk to guide you through the entire Bound Volume Set, all 11 volumes, when you order the complete set. The complete Bound Volume Set, along with a complete index on disk, is yours for ONLY $225! YOU SAVE $138!

Bound Volume Set Eleven years of fascinating history and useful code, along with a complete index on disk, for only $225. Please specify one of the following disk formats: MS-DOS, Macintosh, or Kaypro.
Bound Volumes 1-11
$225 Item #72-0INDX

Bound Volume 11: 1986

The promise of power. With the introduction of the first true 32-bit microprocessors, desktop computers began to rival minis and mainframes in power. Dr. Dobb's Journal of Software Tools covered the changes with special issues on the 68000, parallel processing, artificial intelligence, the 80386, and multitasking. We supported the new chips with assemblers, translators, and other cross-development tools. Other features included a special graphics issue, reviews of Dan Briklin's DEMO program and Jef Raskin's Swyft-Card, and our sixth annual Forth issue. We introduced a new structured languages column, led by Michael Ham and Namir Shammas, while Ray Duncan and Allen Holub continued to provide their own valuable columns.

► $35.75 Item #31-3

To receive all 1986 listings on disk, FREE with your Bound Volume 11 order, specify one of the following disk formats: MS-DOS, Macintosh, or Kaypro.

► $35.75 Item #31-3LIST

Bound Volume Index The index of the complete Bound Volume series is also available on disk in MS-DOS, Macintosh or Kaypro formats for only $9.95
Bound Volume Index
Item #80-1 ► **$9.95**

Bound Volume 1: 1976

The working notes of a technological revolution. Before there was Apple, *DDJ* put a programming language on the first microcomputers and became a chronicler and instrument of the microcomputer revolution.

► $30.75 Item #13-5

Bound Volume 2: 1977

Running light without overbyte. By year two the formula was clear: serious technical questions handled with a minimum of reverence, much source code, and a commitment to tight coding.

► $30.75 Item #16-X

Bound Volume 3: 1978

The roots of Silicon Valley growth. The S-100 bus was hashed out in *DDJ's* pages. Steve Wozniak and others published in *DDJ* code that would help build an industry.

► $30.75 Item #17-8

Bound Volume 4: 1979

In the midst of the gold rush. Three years before IBM moved in, the neighborhood was less civilized. *DDJ* published a gold mine of tips, tricks, and algorithms.

► $30.75 Item #14-3

Bound Volume 5: 1980

C and CP/M. 1980 saw an all-CP/M issue, including Gary Kildall's history of CP/M, and Ron Cain's original Small-C compiler.

► $30.75 Item #18-6

Bound Volume 6: 1981

The first of Forth. This was the year *DDJ* launched its first Forth issue and "Dr. Dobb's Clinic." Plus: PCNET, the Conference Tree, and 6809 Tiny BASIC.

► $30.75 Item #19-4

Bound Volume 7: 1982

Legitimacy. *DDJ* observed the IBM phenomenon, reviewed MS-DOS and CP/M-86, and looked forward to fifth-generation computers.

► $35.75 Item #20-8

Bound Volume 8: 1983

Power tools. Professional software development on a PC was getting easier; *DDJ* helped, with Small-C, the RED editor, and an Ada subset.

► $35.75 Item #00-3

Bound Volume 9: 1984

Shaping things to come. In 1984 *DDJ* examined new programming environments: Prolog, expert systems, Modula-2, and a $49.95 Pascal. Plus Allen Holub's GREP, UNIX internals, and two encryption systems.

► $35.75 Item #08-9

Bound Volume 10: 1985

The year of living dangerously. In 1985, iconoclastic *DDJ* beat Apple to the goal of adding more memory, a SCSI port, and a hard disk to the Macintosh. Also: powerful software tools in C, Modula-2, Forth, Pascal, assembly language, and Prolog.

► $35.75 Item #21-6

Order Form

ORDER NOW! ORDER NOW! ORDER NOW! ORDER NOW! ORDER NOW! ORDER NOW!

NAME ______________________________
(Please use street address, not P.O. Box)

ADDRESS ______________________________

CITY ______________ STATE __________ ZIP __________

DAY PHONE ______________________________

In Calif. 800-356-2002

☐ Check enclosed. Make payable to:
M&T Publishing, Inc., 501 Galveston Dr., Redwood City, CA 94063

☐ Charge My:

☐ Visa ☐ MasterCard ☐ American Express.

Name on card ______________________________

Account No. ______________ Expiration Date __________

Signature ______________________________

For disk orders, please indicate format. Refer to ad for standard format availability for each product.

☐ **MS/DOS** ☐ **Amiga**
☐ **Macintosh** ☐ **Atari 520st**
☐ **Apple II**

CP/M
☐ Kaypro ☐ Osborne
☐ 8″ SS/SD ☐ Apple
☐ Zenith Z-100 DS/DD

For Small-Windows , indicate:
☐ Microsoft C version 4.0 compiler
☐ Small-C Compiler
☐ Lattice-C Compiler
☐ Turbo C

QUANTITY	ITEM #	DESCRIPTION	UNIT PRICE	TOTAL PRICE

SUB-TOTAL __________

CA residents must add applicable sale tax on merchandise total ________ %

SALES TAX __________

Shipping must be included with order. See rates below.

SHIPPING __________

TOTAL ORDER __________

In U.S. For Bound Volumes, add $3.25 per book. Add $9.25 for Special C Packages. For other books and disks, add $2.25 per item. *Outside U.S.* For Bound Volumes, add $6.25 per book surface mail. Add $18 surface mail for Special C Packages. For other books and disks, add $5.25 per item surface mail. Foreign airmail rates available on request. For Fast Service and reduced shipping costs, Germans may order direct from: Markt & Technik, Buchverlag, Hans-Pinsel-Strasse 2, 8013 Haar bei München. Call Germany 089-4613-383 or 089-4613-711 for prices in Deutsch Marks.

3132

M&T Books & Software

ORDER FORM

Please fold along fold-line and staple or tape closed.

No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

First Class Permit No. 871 Redwood City, CA

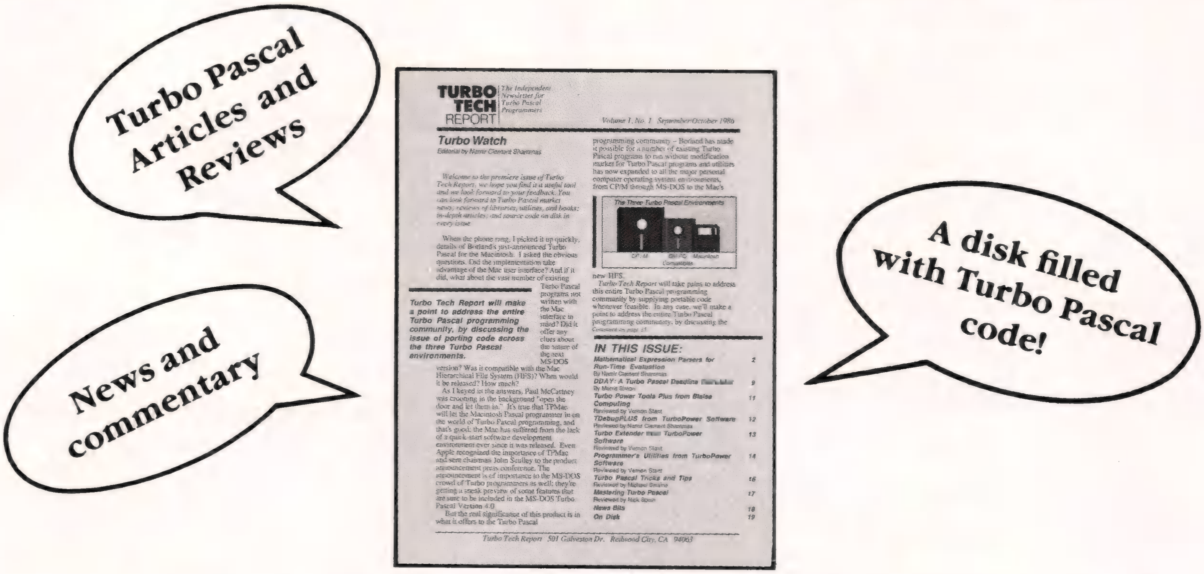
Postage Will Be Paid By Addressee

M&T Books & Software Tools

501 GALVESTON DRIVE
REDWOOD CITY, CA 94063

Please fold along fold-line and staple or tape closed.

Turbo Tech Report Speaks Your Language.



The newsletter/disk publication for Turbo Pascal® users

Are you looking for powerful utilities written in Turbo Pascal that you can use to develop software or incorporate into your programs? Are you interested in improving and expanding your Turbo Pascal programming skills?

Then you deserve a subscription to *Turbo Tech Report*, the bimonthly newsletter/disk publication from the publishers of *Dr. Dobb's Journal* and *Micro/Systems Journal*. Each issue delivers more than 250K of Turbo Pascal source code programs on disk, and 24+ pages of articles, Turbo Pascal software and book reviews, and analysis and commentary. It's the only publication delivering such focused technical articles with code on disk. Each valuable issue contains:

- ***Articles*** on topics like speedy 3D graphics, mathematical expression parsers, creating global gotos, memory resident and AI applications and more—all written by Turbo experts.
- ***Reviews*** of the latest Turbo Pascal software programs from companies like Borland International, Blaise Computing, Media Cybernetics, Nostradamus, and more!
- ***News and commentary*** detailing the latest products and developments in the Turbo Pascal programming community.
- ***A disk filled with Turbo Pascal code!*** You'll get the Turbo Pascal utilities and routines discussed in the newsletter's articles, as well as applications developed by Turbo users from around the world. You'll receive programs that make labels, generate menus, provide faster screen access, transfer files, and more!

If you're an expert Turbo Pascal programmer or a novice interested in expanding your Turbo skills, you need a publication that speaks your language: *Turbo Tech Report*. Subscribe today at the special price of just $99—that's 33% off the regular price of $150. To order by credit card, call toll-free 1-800-533-4372. Or mail the coupon with your payment to *Turbo Tech Report*, 501 Galveston Drive, Redwood City, CA 94063.

—Yes! I want a one-year subscription to *Turbo Tech Report* (6 issues with 6 disks) for $99.

Format: ☐ PC/MS-DOS ☐ Macintosh

CP/M: ☐ Kaypro ☐ Osborne ☐ Apple

PAYMENT MUST ACCOMPANY ALL ORDERS

☐ Check/money order enclosed.

☐ Charge my: ___ VISA ___ M/C ___ AmExp.

Card # ______________________ Exp. __________

Signature ______________________

Name ______________________

Address ______________________

City ______________ State ________ Zip ________

Orders outside U.S.: add $30.

CIRCLE 119 ON READER SERVICE CARD

TRUE
MULTITASKING
With
MultiDos Plus
"multitasking for the IBM-PC."

Ideal for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in most languages.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
  - Intertask message communication. Send/receive/check message present on 64 message queues.
  - Task control by means of semaphores. Get/release/check semaphores.
  - Change priority-256 priority levels.
  - Suspend task for specified interval.
  - Spawn and terminate external and internal tasks.
  - Disable/enable multitasking.
  - and more!
- Independent foreground/background displays.
- Access to DOS while applications are running.

Hardware/Software Requirements

IBM PC/XT/AT or true clone. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

only: **$24.95** *OR* **$99.95** with source code

Outside USA add $5.00 shipping and handling.
Visa and Mastercard orders only call toll-free: 1-800-872-4566, ext. 350., or send check or money order to:

NANOSOFT
13 Westfield Rd, Natick, MA 01760
MA orders add 5% sales tax.

CIRCLE 309 ON READER SERVICE CARD

Listing Three *(Text begins on page 46.)*

```
/*
 *
 *                              FINDCMD
 *
 *                         MODULE: STATE.C
 *
 *              COPYRIGHT (C) 1987 by Charles F. Bowman
 *                         All Rights Reserved.
 *
 */


/*
 *      This module contains the routines nesessary
 *      to implement the state machine
 */
#include        <stdio.h>

#include        "findcmd.h"

static  int     tos = -1;
static  int     pat[ 100 ];
static  struct  stk     stk[ 100 ];

/*
 *      INITS: initialize the state machine (transition table)
 */
inits( p )
char    *p;
{
        register int    i;

        i = 1;
        pat[0] = PATBEG;
        while( *p != NIL ){
                /*
                 *      Add each char in pattern to state array
                 */
                switch( *p ){
                case '?':
                        pat[i] = QUEST;
                        break;
                case '*':
                        pat[i] = ASTER;
                        break;
                default:
                        pat[i] = *p;
                        break;
                }
                p++;
                i++;
        }
        pat[i] = PATEND;

        return( 0 );
}

/*
 *      STATE: driving routine for the state machine;
 *              performs the actual pattern matching.
 */
state( n )
char    *n;
{
        register int    state;
        char    *ptr;

        ptr = n;
        tos = -1;
        state = 0;
        for(;;){                                /* Forever */

                switch( pat[state] ){
                case PATBEG:                    /* Begin state */
                        break;
```

```
        case ASTER:                         /* Wild card */
                if( *(ptr+1) != NIL ){
                        /*
                         *      Save machine state
                         */
                        PUSH( state-1, ptr+1 );
                }
                while( (*ptr != pat[state+1]) && (*ptr != NIL) ){
                        /*
                         *      Skip non-matching chars
                         *      up to end of string
                         */
                        ptr++;
                }
                break;

        case PATEND:                        /* End state */
                if( *ptr == NIL ){
                        /*
                         *      Match!
                         */
                        return( 1 );
                } else if( TOS ){
                        /*
                         *      No match - restore saved state
                         */
                        POP( state, ptr );
                } else {
                        /*
                         *      No match!
                         */
                        return( 0 );
                }
                break;

        case QUEST:                         /* Any 1 character */
                ptr++;
                break;

        default:
                if( *ptr != pat[state] ){
                        if( TOS ){
                                /*
                                 *      Restore saved state
                                 */
                                POP( state, ptr );
                        } else {
                                /*
                                 *      Fail - no match!
                                 */
                                return( 0 );
                        }
                } else {
                        /*
                         *      Equal - move on
                         */
                        ptr++;
                }
                break;
        }
        state++;                            /* Next state */
    }
}
```

End Listing Three

Listing Four

```
/*
 *
 *                      FINDCMD
 *
 *                  MODULE: FINDCMD.H
 *
 *          COPYRIGHT (C) 1987 by Charles F. Bowman
 *                  All Rights Reserved.
 *
 */
```

(continued on next page.)

UNIFORTH™

NOW: a FORTH System that meets *your* needs!

Available for most personal computers and DEC minis, this FORTH-83 language from an established company will meet the needs of the *novice* and the *expert* alike!

Beginners will enjoy the clear style of the Users Guide. The video editor is easy to use, and the debug/decompile utilities make programming a snap. Leave questions on our Bulletin Board Service for quick answers. Our optional telecommunications package provides superb modem access.

Advanced Programmers will get more details of the system from the comprehensive Programmers Guide. Optimized code and full memory space utilization make **UNIFORTH** one of the fastest FORTHs for large applications. Use the complete macro assembler for time-critical routines. Text file, operating system, graphics, sound, and string support are included.

Engineers will use the multitasking and high-level interrupt features to handle simultaneous functions. We have ROM-based systems for many popular single-board computers, or you can use one of our optional cross-compilers for one of a dozen processors to generate your own systems. Need help? We offer custom programming, classes and other consulting services.

Scientists will appreciate the full floating-point support. Optional utilities include a 2-D plotting package, astronomical programs and catalogs, image processing and artificial intelligence.

Public-Domain Samplers are available for many systems.

Call or write for our **free catalog.** We guarantee that you'll enjoy **UNIFORTH, or your money back!**

Unified Software Systems
P.O. Box 21294
Columbus, OH 43221
(614) 459-7735
BBS: (614) 459-7736
(300/1200 baud, 24 hrs.)

CIRCLE 188 ON READER SERVICE CARD

Listing Four *(Listing continued, text begins on page 46.)*

```
/*
 *      Header file for FINDCMD.C
 */
#define NIL     '\0'
#define FLDSEP  ';'                     /* dir separator in path */
#define MAXPATH 250

/*
 *      Machine states
 */
#define ASTER   128
#define QUEST   129
#define PATBEG  130
#define PATEND  131

/*
 *      DOS file attribute bits
 */
#define RDONLY  0x01                    /* Read only file */
#define HIDDEN  0x02                    /* Hidden file */
#define SYSTEM  0x04                    /* System file */
#define DIRECT  0x10                    /* Directory file */
#define ARCHIV  0x20                    /* Archive bit */

#define SA_SIZE(foo)    (sizeof(foo)/sizeof(foo[0]))

/*
 *      Macros for stack manipulation
 */
#define TOS             (tos == -1 ? 0 : 1)
#define POP(x,y)        {x = stk[tos].state; y = stk[tos].ptr; tos--; }
#define PUSH(x,y)       { tos++; stk[tos].state = (x); stk[tos].ptr = (y); }

char    *slcase();

/*
 *      Internal DOS date structure
 */
struct dosdate {
        unsigned sec    : 5;            /* Second (intervals of 2) */
        unsigned min    : 6;            /* Minutes */
        unsigned hour   : 5;            /* Hours */
        unsigned day    : 5;            /* Day of month */
        unsigned month  : 4;            /* Month of year */
        unsigned year   : 7;            /* Year since 1980 */
};


/*
 *      DOS FCB / DIR ENTRY - Set to DTA
 */
struct  dirent  {
        char    dinfo[ 21 ];
        char    dattr;
        struct  dosdate ddate;
        long    dsize;
        char    dname[ 13 ];
};
/*
 *      Used to print attribute bits of files
 */
struct attribs {
                int     val;
                int     chr;
};

/*
 *      Used to save an restore machine state
 */
struct  stk     {
        int     state;
        char    *ptr;
};
```

End Listings

SAS Institute Inc. Announces

Lattice C Compilers for Your IBM Mainframe

Two years ago...

SAS Institute launched an effort to develop a subset of the SAS® Software System for the IBM Personal Computer. After careful study, we agreed that C was the programming language of choice. And that the Lattice® C compiler offered the quality, speed, and efficiency we needed.

One year ago...

Development had progressed so well that we expanded our efforts to include the entire SAS System on a PC, written in C. And to insure that the language, syntax, and commands would be identical across all operating systems, we decided that all future versions of the SAS System—regardless of hardware—would be derived from the same source code written in C. That meant that we needed a C compiler for IBM 370 mainframes. And it had to be good, since all our software products would depend on it.

So we approached Lattice, Inc. and asked if we could implement a version of the Lattice C compiler for IBM mainframes. With Lattice, Inc.'s agreement, development began and progressed rapidly.

Today...

Our efforts are complete—we have a first-rate IBM 370 C compiler. And we are pleased to offer this development tool to you. Now you can write in a single language that is source code compatible with your IBM mainframe and your IBM PC. We have faithfully implemented not only the language, but also the supporting library and environment.

Features of the Lattice C compiler for the 370 include:

- **Generation of reentrant object code.** Reentrancy allows many users to share the same code. Reentrancy is not an easy feature to achieve on the 370, especially if you use non-constant external variables, but we did it.
- **Optimization of the generated code.** We know the 370 instruction set and the various 370 operating environments. We have over 100 staff years of assembler language systems experience on our development team.
- **Generated code executable in both 24-bit and 31-bit addressing modes.** You can run compiled programs above the 16 megabyte line in MVS/XA.
- **Generated code identical for OS and CMS operating systems.** You can move modules between MVS and CMS without even recompiling.
- **Complete libraries.** We have implemented all the library routines described by Kernighan and Ritchie (the informal C standard), and all the library routines supported by Lattice (except operating system dependent routines), plus extensions for dealing with 370 operating environments directly. Especially significant is our byte-addressable Unix®-style I/O access method.
- **Built-in functions.** Many of the traditional string handling functions are available as built-in functions, generating in-line machine code rather than function calls. Your call to move a string can result in just one MVC instruction rather than a function call and a loop.

In addition to mainframe software development, you can also use our new cross-compiler to develop PC software on your IBM mainframe. With our cross-compiler, you can compile Lattice C programs on your mainframe and generate object code ready to download to your PC.

With the cross-compiler, we also offer PLINK86™ and PLIB86™ by Phoenix Software Associates Ltd. The Phoenix link-editor and library management facility can bind several compiled programs on the mainframe and download immediately executable modules to your PC.

Tomorrow...

We believe that the C language offers the SAS System the path to true portability and maintainability. And we believe that other companies will make similar strategic decisions about C. Already, C is taught in most college computer science curriculums, and is replacing older languages in many. And almost every computer introduced to the market now has a C compiler.

C, the language of choice...

C supports structured programming with superior control features for conditionals, iteration, and case selection. C is good for data structures, with its elegant implementation of structures and pointers. C is conducive to portable coding. It is simple to adjust for the size differences of data elements on different machines.

Continuous support...

At SAS Institute, we support all our products. You license them annually; we support them continuously. You get updates at no additional charge. We have a continuing commitment to make our compiler better and better. We have the ultimate incentive—all our software products depend on it.

For more information...

Complete and mail the coupon today. Because we've got the development tool for your tomorrow.

SAS Institute Inc.
SAS Circle, Box 8000
Cary, NC 27511-8000
Telephone (919) 467-8000 x 7000

I want to learn more about:

- ☐ the C compiler for MVS software developers
- ☐ the C compiler for CMS software developers
- ☐ the cross-compiler with PLINK86 and PLIB86

today...so I'll be ready for tomorrow.

Please complete or attach your business card.

Name ______________________

Title ______________________

Company ______________________

Address ______________________

City ______________ State ________ ZIP ________

Telephone ______________________

Mail to: SAS Institute Inc., Attn: CC, SAS Circle, Box 8000, Cary, NC, USA. 27511-8000. Telephone (919) 467-8000, x 7000

DDJ 10/87

SAS is the registered trademark of SAS Institute Inc., Cary, NC, USA. Lattice is the registered trademark of Lattice, Inc. PLINK86 and PLIB86 are trademarks of Phoenix Software Associates Ltd. UNIX is the trademark of AT&T. Copyright © 1985 by SAS Institute Inc. Printed in the USA.

Listing One *(Text begins on page 140.)*

```
Listing 1.  QuickBASIC library to implement opaque matrices.

' QuickBASIC implementation of an opaque numeric matrix
' Matrix is stored as arrays of columns
' OPTION BASE 0 must be used, although the row/column indices
' start at one.

SUB InitMat(Mat#(1), Max.Row%, Max.Col%) STATIC
' Initialize matrix

Mat#(0) = Max.Row% + Max.Col% / 1000

FOR I% = 1 TO UBound(Mat#)
   Mat#(I%) = 0
NEXT I%

END SUB ' CreateMat

SUB StoreElem(Mat#(1), Row%, Col%, Elem#, OK%) STATIC

' Store Elem# in matrix position (Row%,Col%)
' OK% is zero if error has occurred, -1 if operation was done

STATIC I%, MaxR%, MaxC%

MaxR% = INT(Mat#(0))
MaxC% = 1000 * (Mat#(0) - MaxR%)

IF (MaxR% < Row%) OR (MaxC% < Col%) OR (Row% < 1) OR (Col% < 1) THEN
   OK% = 0 ' Bad row or column numbers.
   EXIT SUB
END IF

OK% = -1

' Calculate index
I% = Row% + (Col% - 1) * MaxR%
' for the arrays of rows representation use
'    I% = Col% + (Row% - 1) * MaxC%


' Store element
Mat#(I%) = Elem#

END SUB ' StoreElem

SUB RecallElem(Mat#(1), Row%, Col%, Elem#, OK%) STATIC

' Recall Elem# in matrix position (Row%,Col%)
' OK% is zero if error has occurred, -1 if operation was done

STATIC I%, MaxR%, MaxC%

MaxR% = INT(Mat#(0))
MaxC% = 1000 * (Mat#(0) - MaxR%)

IF (MaxR% < Row%) OR (MaxC% < Col%) OR (Row% < 1) OR (Col% < 1) THEN
   OK% = 0 ' Bad row or column numbers.
   EXIT SUB
END IF

OK% = -1

' Calculate index
I% = Row% + (Col% - 1) * MaxR%
' for the arrays of rows representation use
'    I% = Col% + (Row% - 1) * MaxC%

' Recall element
Elem# = Mat#(I%)

END SUB ' RecallElem
```

End Listing One

Listing Two

```
Listing 2.  True BASIC module that implements an array-based binary tree.

MODULE  Binary_Tree

! TRUE BASIC module that implements a single binary tree
! Copyright (c) 1987  Namir Clement Shammas

DECLARE DEF NIL, TRUE, FALSE
SHARE Left(1), Right(1), Node_Count, Num_Nodes, Bin_Tree$(1)


!------------ Module initialization ---------
LET Num_Nodes = 0

!----------- local functions -----------
DEF NIL = MAXNUM
DEF TRUE = 1
DEF FALSE = 0


SUB Initialize(Item$)
! Subroutine to initialize the binary tree

LET Num_Nodes = 1
LET Tree_Size = 1
LET Bin_Tree$(1) = Item$
LET Left(1) = NIL
LET Right(1) = NIL

END SUB


SUB Search(Item$, Found, Index)
! Search for Item$ and return Index if found.
```

(continued on page 113)

FORTH
LEADING THE NEW GENERATION OF COMPUTER LANGUAGES

The Forth Interest Group (FIG) is the association of programmers, managers and engineers who create practical, Forth-based solutions to real-world problems. Their uses of Forth include:

- Instrument Control
- Data Acquisition
- Stand-alone Devices
- Expert Systems
- Business Accounting
- Graphics Systems

Each FIG member receives the bimonthly magazine *Forth Dimensions*.

Its tutorials, innovative techniques, and extensive program listings are published exclusively for FIG members, and are refereed by premier experts in the Forth community.

The 9th Annual Forth Convention will be held at the Red Lion Inn in San Jose, California on November 13 and 14, 1987. Each year, Forth enthusiasts attend the program of lectures and panel discussions by the field's most respected experts, and exhibits by Forth vendors as well.

FORTH INTEREST GROUP
fig

To join FIG or for further information, call us at **(408) 277-0668** or write P. O. Box 8231, San Jose, CA 95155, USA.

CIRCLE 231 ON READER SERVICE CARD

Amazing COMPUTING™
Your Original AMIGA™ Monthly Resource

FEATURING

- Complete Amiga Hardware and Software reviews
- A vast and growing library of over 110 PDS Disks
- Solid and informative for both the advanced and beginning Amiga User
- Understandable program listings and tools
- Step by Step Hardware projects

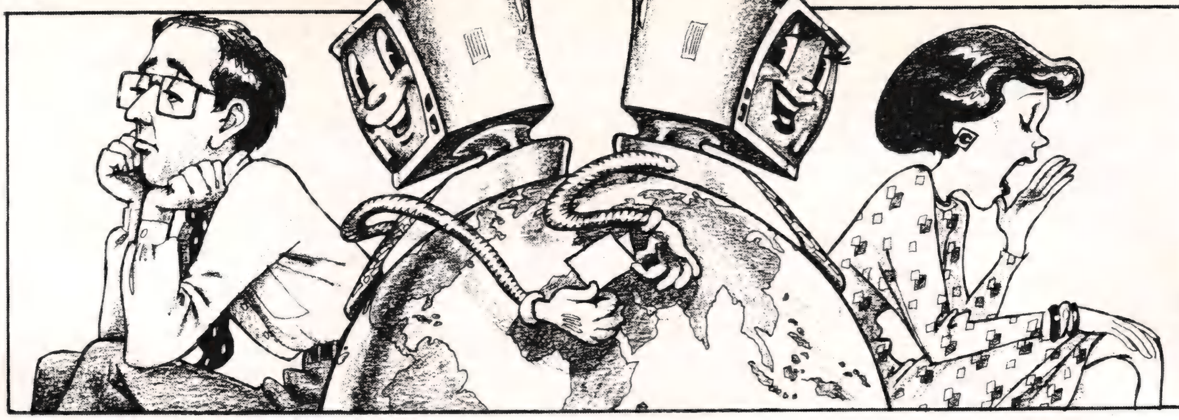
Amiga Users have made Amazing Computing™ the longest running Monthly magazine dedicated to the Commodore Amiga. If you are searching for Amiga technical information that is both current and comprehensive, then be amazed by the pioneer Amiga Magazine, Amazing Computing - your Original AMIGA Monthly Resource.

YES, Amaze Me! I have enclosed $24.00 U.S. ($30.00 Canada & Mexico, $35.00 Overseas) in check or money order (U.S. funds drawn on a U.S. bank) to:

PiM Publications, Inc.
P.O. Box 869-DD
Fall River, MA 02722

Name ______
Address ______
City ______ St ______ Zip ______

CIRCLE 136 ON READER SERVICE CARD



Ten Reasons Not to Use PeaceNet

1. I don't like working with others
PeaceNet is a computer network and communication system for people who believe that global planning and cooperation are necessary to reverse a trillion-dollar-per-year arms race; it is linking users throughout the United States and in over 70 other countries.

2. I've got all the information I'll ever need
PeaceNet is for those who appreciate that information is always growing and changing; its bulletin boards, conferences, and databases provide information about everything from Central America to Star Wars.

3. I love playing phone tag
PeaceNet's electronic mail system renders those endless conversations with secretaries and answering machines obsolete.

4. I don't know how to use my computer
PeaceNet helps novices with simple, entertaining manuals and round-the-clock staff for answering their questions.

5. I enjoy copying, labeling, and stamping letters
PeaceNet enables you to send messages to hundreds of other users with one simple command.

6. I've got plenty of money to waste on postage and phone bills
PeaceNet is for people who want to save money; it lets you send documents across the world faster than Federal Express™ for pennies per page.

7. I don't mind getting action alerts a week late
PeaceNet does mind and can help your organization send out time-urgent alerts instantly.

8. I don't have the right kind of computer equipment
PeaceNet is available to anyone with a computer terminal and a modem.

9. An effective peace movement isn't worth 50 cents a day
PeaceNet users disagree.

10. It's all hopeless, anyway
Then why read this magazine when Modern Wrestling would suffice?

Nearly a thousand people and fifty groups are already using PeaceNet, including Beyond War, the National Freeze Campaign, and Nuclear Times. If you want to join them in an unprecedented international dialogue for peace, write or call us today for details.

PeaceNet: The First Global Computer Network for Peace
1918 Bonita, Berkeley, CA 94704 (415) 486-0264

Listing Two *(Listing continued, text begins on page 140.)*

```
LET Found = FALSE
LET Index = 1

DO WHILE (Index <> NIL) AND (Found = FALSE)
   IF Bin_Tree$(Index) = Item$ THEN
      LET Found = TRUE
   ELSE
      IF Bin_Tree$(Index) < Item$ THEN
         LET Index = Right(Index)
      ELSE
         LET Index = Left(Index)
      END IF
   END IF
LOOP

END SUB


SUB Insert(Item$)
! Insert Item$ in the "dynamic" binary tree structure

LET Num_Nodes = Num_Nodes + 1

IF Num_Nodes > Tree_Size THEN
   LET Tree_Size = Num_Nodes
   MAT REDIM Bin_Tree$(Tree_Size), Left(Tree_Size), Right(Tree_Size)
END IF

LET Index = 1
LET Found = FALSE

DO WHILE Index <> NIL
   IF Bin_Tree$(Index) < Item$ THEN
      IF Right(Index) <> NIL THEN
         LET Index = Right(Index)

      ELSE
         LET Right(Index) = Num_Nodes
         LET Index = NIL
      END IF
   ELSE
      IF Left(Index) <> NIL THEN
         LET Index = Left(Index)
      ELSE
         LET Left(Index) = Num_Nodes
         LET Index = NIL
      END IF
   END IF
LOOP

LET Bin_Tree$(Num_Nodes) = Item$
LET Right(Num_Nodes) = NIL
LET Left(Num_Nodes) = NIL

END SUB

END MODULE
```

End Listing Two

Listing Three

```
Listing 3.  Pascal code for emulating opaque complex data types.


TYPE
    Opaque_Complex_type = ^Opaque_Complex_type_record;

    { record type is deliberately empty }
    Opaque_Complex_type_record = RECORD
                                 END;
```

(continued on next page)

COMBINE THE RAW POWER OF FORTH WITH THE CONVENIENCE OF CONVENTIONAL LANGUAGES

HS/FORTH

Yes, Forth gives you total control of your computer, but only HS/FORTH gives you implemented functionality so you aren't left hanging with "great possibilities" (and lots of work!) With over 1500 functions you are almost done before you start!

WELCOME TO HS/FORTH, where megabyte programs compile at 10,000 lines per minute, and execute faster than ones built in 64k limited systems. Then use AUTOOPT to reach within a few percent of full assembler performance — not a native code compiler linking only simple code primitives, but a full recursive descent optimizer with almost all of HS/FORTH as a useable resource. Both optimizer and assembler can create independent programs as well as code primitives. The metacompiler creates threaded systems from a few hundred bytes to as large as required, and can produce ANY threading scheme. And the entire system can be saved, sealed, or turnkeyed for distribution either on disk or in ROM (with or without BIOS).

HS/FORTH is a first class application development and implementation system. You can exploit all of DOS (commands, functions, even shelled programs) and link to .OBJ and .LIB files meant for other languages *interactively!*

I/O is easier than in Pascal or Basic, but much more powerful — whether you need parsing, formatting, or random access. Send display output through DOS, BIOS, or direct to video memory. Windows organize both text and graphics display, and greatly enhance both time slice and round robin multitasking utilities. Math facilities include both software and hardware floating point plus an 18 digit integer (finance) extension and fast arrays for all data types. Hardware floating point covers the full range of trig, hyper and transcendental math including complex.

Undeniably the most flexible & complete Forth system available, at any price, with no expensive extras to buy later. Compiles 79 & 83 standard programs. Distribute metacompiled tools, or turnkeyed applications royalty free.

HS/FORTH (complete system):	$395.
HS/FORTH: Tutorial & Ref (500 pg)	$ 59.
Forth: Text & Reference (500 pg)	$ 22.
HS/FORTH Glossary	$ 10.
GIGAFORTH Option (Beta release)	$245.

(Native Mode from SOFTMILLS, INC.)

VISA Visa Mastercard MasterCard

HARVARD SOFTWORKS
PO BOX 69
SPRINGBORO, OH 45066
(513) 748-0390

CIRCLE 132 ON READER SERVICE CARD

Listing Three *(Listing continued, text begins on page 140.)*

```
    Actual_Complex_type = ^Actual_Complex_type_record;

    Actual_Complex_type_record = RECORD
                                   Reel,
                                   Imag : REAL;
                                 END;

    Convert_Complex = RECORD
                        CASE BOOLEAN OF
                            TRUE  : (Opaque : Opaque_Complex_type);
                            FALSE : (Actual : Actual_Complex_type)
                        END;


FUNCTION Convert_Opaque_to_Actual( Opaque_Complex : Opaque_Complex_type ) :
                                   Actual_Complex_type;

VAR Transfer : Convert_Complex;

BEGIN
    Transfer.Opaque := Opaque_Complex;
    Convert_Opaque_to_Actual := Transfer.Actual
END; { Convert_Opaque_to_Actual }



FUNCTION Convert_Actual_to_Opaque( Actual_Complex : Actual_Complex_type ) :
                                   Opaque_Complex_type;

VAR Transfer : Convert_Complex;

BEGIN
    Transfer.Actual := Actual Complex;
```


Publication Quality Scientific Graphics

GraphiC 4.0 in color

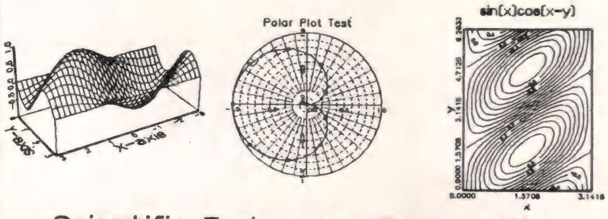
Over 125 C routines make scientific plotting easy

- → linear, log, & polar plots
- → bar charts & Smith charts
- → contour plots with labels
- → 3-D curves, 3-D surfaces
- → 4 curve types, 8 markers, errorbars
- → 14 fonts, font editor
- → unlimited levels of super/sub scripts
- → 4096 x 3120 resolution in 16 colors on EGA, Tecmar, Sigma boards
- → zoom, pan, window and merge plots
- → high resolution printer dumps

SOURCE INCLUDED for *personal* use only

$350. Demo $8

256k, IBM, AT&T, Corona PCs, DOS 2.xx, 3.xx
Most boards, printers, and plotters supported
Microsoft, Lattice, DeSmet, Aztec, C86 compilers



Scientific Endeavors Corporation
Route 4, Box 79 Kingston, TN 37763 (615) 376-4146

CIRCLE 210 ON READER SERVICE CARD

CANADA'S SOURCE FOR C

- Canadian Sales
- Canadian Service
- Canadian Technical Support
- Canadian Product Knowledge

We specialize in programming & development software

LIFEBOAT • LATTICE • GREENLEAF • PHOENIX
SOFTCRAFT • MICROSOFT • BLAISE • ESSENTIAL
AGE OF REASON • DESMET • AZTEC
MARK WILLIAMS • GIMPEL • ROUNDHILL • GSS
HALO • FAIRCOM • RAIMA • INTEL • etc. • etc. •

Call for full price list—Dealer enquiries welcome

We know our products—we use them!

SCANTEL SYSTEMS LTD.
801 York Mills Rd., Don Mills, Ont., M3B 1X7
(416) 449-9252

CIRCLE 391 ON READER SERVICE CARD

```
    Convert_Actual_to_Opaque := Transfer.Opaque
END; { Convert_Actual_to_Opaque }



FUNCTION Real_Imag_Complex(Re, Im : REAL) : Opaque_Complex_type;
{ Convert from Real/Imaginary numbers to opaque complex numbers }
VAR Transfer : Actual_Complex_type;

BEGIN
    NEW(Transfer);
    Transfer^.Reel := Re;
    Transfer^.Imag := Im;
    Real_Imag_Complex:= Convert_Actual_to_Opaque(Transfer);
END; { Real_Imag_Complex }


FUNCTION Polar_Complex(Angle, Modulus : REAL) : Opaque_Complex_type;
{ Convert from polar coordinates to opaque complex numbers }
VAR Transfer : Actual_Complex_type;

BEGIN
    NEW(Transfer);
    Transfer^.Reel := Modulus * SIN(Angle);
    Transfer^.Imag := Modulus * COS(Angle);
    Real_Imag_Complex:= Convert_Actual_to_Opaque(Transfer);
END; { Polar_Complex }



PROCEDURE Get_Real_Imag(MyComplex : Opaque_Complex_type;
                        VAR Re, Im : REAL { output});
{ Convert opaque complex numbers into Real/Imaginary components }
VAR Transfer : Actual_Complex_type;
```

(continued on next page)

INTELLIGENT PROGRAMMING

A Selection of Books from Springer-Verlag

New edition!
PROGRAMMING IN PROLOG
THIRD EDITION
W.F. Clocksin and **C.S. Mellish**
Now in its third revised edition, **Programming in Prolog** includes an account of up-to-date programming techniques using accumulators and difference structures, new information on syntax errors, and operator precedences that are compatible with the most widely used implementations. Also contains substantial revisions and improvements in presentation.
1987/281 pp/7 illus/Softcover $19.95
ISBN 0-387-17539-3

Fully updated edition!
CATALOGUE OF ARTIFICIAL INTELLIGENCE TOOLS
SECOND REVISED EDITION
Edited by **A. Bundy**
Revised to keep pace with the rapid changes taking place in AI, the **Catalogue of AI Tools** contains paragraph-length descriptions of current AI techniques and software. Nearly 300 items covering the areas of Automatic Programming, Computer Architecture, Expert Systems, Logic Programming, Natural Language, and many more.
1986/168 pp/Softcover $27.50 ISBN 0-387-16893-1
(Symbolic Computation)

PASCAL USER MANUAL AND REPORT
Revised for ISO Pascal Standard
THIRD EDITION
K. Jensen and **N. Wirth**
The Pascal classic has been up-dated to incorporate the ISO standard. Written by the designer of the language, this new edition retains the rigor and precision that has made the **Pascal User Manual and Report** a definitive reference source for nearly a decade.
1985/266 pp/76 illus/Softcover $17.50
ISBN 0-387-96048-1

THE SCIENCE OF PROGRAMMING
D. Gries
Now in a handy paperback edition, **The Science of Programming** is the first book that not only describes basic programming principles, but which leads the reader step-by-step through the application of these principles. The beginner and the experienced programmer will find this book to be full of useful information.
1981/366 pp/Hardcover $29.00 ISBN 0-387-90641-X
Softcover $19.00 ISBN 0-387-96480-0
(Texts and Monographs in Computer Science)

To order these and other Springer-Verlag computer science titles, send a check or money order (plus $2.50 for shipping) to: Springer-Verlag New York, Inc., Attn: G. Kiely, 175 Fifth Avenue, New York, NY 10010. NY, NJ, and CA residents please add sales tax. To order by credit card, call TOLL FREE 1-800-526-7254 (in NJ, 201-348-4033).

Springer-Verlag
New York Berlin Heidelberg Vienna
London Paris Tokyo

CIRCLE 236 ON READER SERVICE CARD

Listing Three *(Listing continued, text begins on page 140.)*

```
BEGIN
    Transfer := Convert_Opaque_to_Actual(MyComplex);
    Re := Transfer^.Reel;
    Im := Transfer^.Imag;
END; { Get_Real_Imag }


PROCEDURE Get_Polar(MyComplex : Opaque_Complex_type;
                    VAR Angle, Modulus : REAL { output});
{ Convert opaque complex numbers into polar components }
VAR Transfer : Actual_Complex_type;

BEGIN
    Transfer := Convert_Opaque_to_Actual(MyComplex);
    WITH Transfer^ DO BEGIN
        Modulus := SQRT(SQR(Reel) + SQR(Imag));
        Angle   := Imag / Reel;
    END; { WITH }
END; { Get_Polar }



FUNCTION Add_Complex(C1, C2 : Opaque_Complex_type) : Opaque_Complex_type;


VAR Transfer : Actual_Complex_type;
    Re, Im : REAL;

BEGIN
    { Get first complex number }
    Transfer := Convert_Opaque_to_Actual(C1);
    Re := Transfer^.Reel;
    Im := Transfer^.Imag;
    { Get second complex number }
    Transfer := Convert_Opaque_to_Actual(C2);
    Re := Re + Transfer^.Reel;
    Im := Im + Transfer^.Imag;
    { Update result }
    Transfer^.Reel := Re;
    Transfer^.Imag := Im;
    Add_Complex := Convert_Actual_to_Opaque(Transfer);
END; { Add_Complex }
```

End Listing Three

Listing Four

```
Listing 4.  Modula-2 code for opaque complex data types.


DEFINITION MODULE Complex;

EXPORT QUALIFIED Complex, RealImagComplex, PolarComplex,


TYPE  Complex; (* opaque type *)


PROCEDURE RealImagComplex(Re, Im : REAL) : Complex;
(* Convert from Real/Imaginary numbers to opaque complex numbers *)

PROCEDURE PolarComplex(Angle, Modulus : REAL) : Complex;
(* Convert from polar coordinates to opaque complex numbers *)

PROCEDURE GetRealImag(MyComplex : Complex; VAR Re, Im : REAL (* output *));
(* Convert opaque complex numbers into Real/Imaginary components *)


PROCEDURE GetPolar(MyComplex : Complex; VAR Angle, Modulus : REAL (* output*));
(* Convert opaque complex numbers into polar components *)

PROCEDURE AddComplex(C1, C2 : Complex) : Complex;

END Complex.
```

(continued on page 119)

THE PROGRAMMER'S SHOP

helps save time, money and cut frustrations. Compare, evaluate, and find products.

RECENT DISCOVERY

HUMMINGBOARD 386-Develop 2.6 or 7.9 times faster than a 8 MHZ AT. AT or XT addin board uses dual processors for Speed and Hardware Debugging. 16 MHZ or 20 MHZ. Call about Benchmarks, Trial Program.

AI-Expert System Dev't

Arity Combination Package	PC	$ 979
System - use with C	MS	$ 229
SQL Dev't Package	MS	$ 229
Auto-Intelligence	PC	$ 739
CxPERT - shell for C	MS	$ 295
Experteach - Powerful, samples	PC	$ 339
Exsys	PC	$ 289
Runtime System	PC	$ 469
Level 5	MS	$ 659
Intelligence/Compiler	PC	$ 739
T.I.: PC Easy	PC	$ 435
Personal Consultant Plus	PC	$2589
Personal Consultant Runtime	PC	$ 85
Turbo Expert-Startup(400 rules)	PC	$ 129
Corporate (4000 rules)	PC	$ 359

AI-Lisp

Microsoft Lisp V5.1	MS	$ 159
PC Scheme LISP - by TI	PC	$ 85
Star Sapphire	MS	$ 459
TransLISP - learn fast	MS	$ 79
TransLISP PLUS	MS	$ 125

Others: IQ LISP ($239), IQC LISP ($269)

AI-Prolog

APT - Active Prolog Tutor - build applications interactively	PC	$ 49
ARITY Prolog - full, 4 Meg		
Interpreter - debug, C, ASM	PC	$ 229
COMPILER/Interpreter-EXE	PC	$ 569
Standard Prolog	MS	$ 77
MacProlog Complete	MAC	$ 269
MicroProlog - Prof. Entry Level	MS	$ 85
MicroProlog Prof. Comp./Interp.	MS	$ 439
MPROLOG P550	PC	$ 175
Prolog-86 - Learn Fast	MS	$ 89
Prolog-86 Plus - Develop	MS	$ 199
TURBO PROLOG by Borland	PC	$ 69

Basic

BAS_C - economy	MS	$ 179
BAS_PAS - economy	MS	$ 135
Basic Development System	PC	$ 105
Basic Development Tools	PC	$ 89
Basic Windows by Syscom	PC	$ 95
BetterBASIC	PC	$ 129
Exim Toolkit - full	PC	$ 45
Finally - by Komputerwerks	PC	$ 85
Mach 2 by MicroHelp	PC	$ 55
QBase - screens	MS	$ 79
QuickBASIC	PC	$ **69**
Quick Pak-by Crescent Software	PC	$ 59
Stay-Res	PC	$ 59
True Basic	PC	$ 79
Turbo BASIC - by Borland	PC	$ 69

FEATURE

dB2C Toolkit V 2.0 by Software Connection. 220 + dBIII functions in C source, file handler, windowing, interface to db__VISTA, c-tree, dBCIII, MS, Lattice, Instant C. No Royalties MS $ 289

700 + Programmer's Products

The Programmer's Shop carries every programmer's software product for MSDOS, PCDOS, CPM, Macintosh, Atari, and Amiga systems. We help you choose the best tools for you. Most popular products are in stock, available for quick delivery. We will gladly special order a product for you at no charge — just allow a few extra days for delivery.

Need Cross Compilers, Translators, or the right Fortran compiler? Ask us.

Our Services:

- Programmer's Referral List
- Dealers Inquire
- Compare Products
- Newsletter
- Help find a Publisher
- Rush Order
- Evaluation Literature FREE
- *Over 700 products*
- BBS - 7 PM to 7 AM 617-826-4086 National Accounts Center

C Language-Compilers

AZTEC C86 - Commercial	PC	$499
C86 PLUS - by CI	MS	$359
Datalight C - fast compile, good code, 4 models, Lattice compatible, Lib source. Dev'rs Kit	PC	$ 77
Datalight Optimum - C	MS	$ 99
with Light Tools by Blaise	PC	$168
Lattice C - from Lattice	MS	$269
Let's C Combo Pack	PC	$ 99
Microsoft C 5.0- Codeview	MS	**$275**
Microsoft Quick C	MS	$ 69
Rex - C/86 by Systems & Software - standalone ROM	MS	$695
Turbo C by Borland	PC	$ 69

C Libraries-Files

C Index by Trio/PLUS	MS	$319
BTree by Soft Focus	MS	$ 69
CBTREE - Source, no royalties	MS	$ 99
CTree by Faircom - no royalties	MS	$315
rtree - report generation	PC	$239
dbQUERY - ad hoc, SQL-based	MS	Call
dbVISTA - pointers, network.		
Object only - MSC, LAT,	C86	Call
Source - Single user	MS	Call
dBx - translator to library	MS	$299

C-Screens, Windows, Graphics

C Worthy Interface Lib.	PC	$249
Curses by Aspen Scientific	PC	$109
dBASE Graphics for C	PC	$ 69
ESSENTIAL GRAPHICS - fast	PC	$185
FontWINDOW/PLUS	PC	$229
GraphiC - new color version	PC	$279
Greenleaf Data Windows	PC	$159
w/source	PC	$289
Light WINDOWS/C-Datalight C	PC	$ 79
TurboWINDOW/C - for Turbo C	PC	$ 79
Windows for C - fast	PC	$189
Windows for Data - validation	PC	$319
Vitamin C - screen I/O	PC	$159
View Manager - by Blaise	PC	$199
ZView - screen generator	MS	$139

Atari ST & Amiga

We carry full lines of Manx & Lattice.

Call for a catalog, literature and *solid* value

800-421-8006

THE PROGRAMMER'S SHOP ™

Your complete source for software, services and answers

5-D Pond Park Road, Hingham, MA 02043
Mass: 800-442-8070 or 617-740-2510 8/87

RECENT DISCOVERY

XENIX 386 Toolkit by Santa Cruz. Tools & OS kernel support 4 Gigabyte address space, demand paging, virtual memory paging. Includes MS C, MASM, debugger, file utilities, link kit, runtime library. PC $379

dBASE Language

Clipper compiler	PC	$399
dBASE II	MS	$329
dBase III Plus	PC	$429
dBASE III LANPack	PC	$649
DBXL Interpreter by Word Tech	PC	$139
FoxBASE + - single user	MS	$349
Quicksilver by Word Tech	PC	$439

dBASE Support

dAnalyst	PC	$ 89
dBase Tools for C	PC	$ 65
dBrief with Brief	PC	Call
dBC ISAM by Lattice	MS	$169
Documentor - dFlow superset	MS	$229
Genifer by Bytel-code generator	MS	$279
QuickCode III Plus	MS	$239
R&R Report Generator	MS	$139
Seek-It - Query-by-example	PC	$ 79
Silver Comm Library	MS	$139
Tom Rettig's Library	PC	$ 79
UI Programmer - user interfaces	PC	$249

Fortran & Supporting

50:More FORTRAN	PC	$ 95
ACS Time Series	MS	$465
I/O Pro - screen development	PC	$129
MS Fortran - 4.0, full '77	MS	$279
No Limit - Fortran Scientific	PC	$109
PC-Fortran Tools - xref, pprint	PC	$165
RM/Fortran	MS	Call
Scientific Subroutines - Matrix	MS	$129

Multilanguage Support

BTRIEVE ISAM	MS	$185
BTRIEVE/N-multiuser	MS	$455
GSS Graphics Dev't Toolkit	PC	$375
HALO Development Package	MS	$389
Graphics	PS	$205
Hi-Screen XL - Lotus-style menu, windows	PC	$119
Informix 4GL-application builder	PC	$789
Informix SQL - ANSI standard	PC	$639
NET-TOOLS - NET-BIOS	PC	$129
Opt Tech Sort - sort, merge	MS	$ 99
PANEL	MS	$215
Pfinish - by Phoenix	MS	$229
Polyboost - speed I/O, keyboard	PC	Call
Prime Factor FFT - 8087/287	PC	$145
PVCS Corporate or Personal	MS	Call
QMake by Quilt co.	MS	$ 79
Report Option - for Xtrieve	MS	$109
Screen Sculptor	PC	$ 89
SRMS - new version	MS	$159
Synergy - create user interfaces	MS	$375
Xtrieve - organize database	MS	$199
ZAP Communications - VT 100	PC	$ 89

FEATURE

RTC PLUS by Cobalt Blue. Translate FORTRAN 77 and RATFOR to C except F77I/O, FORTRAN character, and complex expressions. Some DEC F77 extensions. Library C Source. MS $399

Note: All prices subject to change without notice. Mention this ad. Some prices are specials. Ask about COD and POs. Formats: 3" laptop now available, plus 200 others. UPS surface shipping add $3/item.

CIRCLE 133 ON READER SERVICE CARD

NEW PRODUCT!

BASIC DOES DATABASE!

Does db/LIB...?

db/LIB

- **db/LIB™ — performs all the database management tasks for your Microsoft QuickBASIC programs . . . On files made standard by dBASE III.**
- **The db/LIB is a complete set of assembly routines designed to manage Database and B-tree Index Sequential files from programs written in Microsoft's QuickBASIC.**
- **Now you can combine the flexibility of BASIC with the functionality of database management to develop high-speed, full-featured business applications for yourself or your clients.**

db/LIB eliminates the need for BASIC statements such as: Open, File Read, Write, Close, Field. As, Line Input, Print #f, Put, LSet, RSet and replaces them with powerful call routines to let you:

- create relational file structures
- access fields by field name or number
- index on one or many key fields
- find all records that match a key
- have up to 24 DBF or NDX files open
- manage over 4 billion bytes per file
- write programs to interface directly with dBASE III applications

All the power you need to tackle any job!

db/LIB calls look like this:

- CALL GetREC (file%, status%, record#, recdata$)
  → returns into recdata$ the data from record# of the file file%;
  → status% shows successful completion.

Some other db/LIB routines:

- OpenDBF() Creates or opens Datafile
- DefineSTR() Defines data structure
- PutREC() Writes a record to disk
- OpenNDX() Creates or opens Index file
- AddKEY() Puts a key into Index
- GetKEY() Finds a record based on key
- DelKEY() Deletes a key from Index

db/LIB routines are called directly from BASIC

- Links with BCOM or BRUN modules
- Conforms to the QuickBASIC standard CALLing convention

You already know how to use it!

db/LIB makes you more productive by:

- letting your programs work on any datafile
- finding on partial keys, or next, prior
- finding records in large files in seconds
- configuring the pool segment for target machine
- managing internal cache memory buffers
- employing dynamic string allocation
- assisting in conversion of BASIC data files to db/LIB's DBF format
- maintaining unique key, and 'deleted' status
- trapping and diagnosing error conditions

db/LIB takes your BASIC application seriously!

db/LIB utilizes the dBASE III file format, a recognized standard.

- db/LIB is written to the highest specifications to assure top performance.
- Full Documentation and on-disk code are the building blocks of a DBMS.
- **System Requirements:** QuickBASIC 2.01/DOS 2.0+/256K memory

Put our experience to work for you!

If database management is one of your basic needs, then db/LIB should be your next call. db/LIB will perform as described or your money back. **$139**

(800) 992-3383
In California Call
(818) 985-3383
9:00 am - 5:00 pm
Pacific Time
VISA, MasterCard Accepted

AJS PUBLISHING, INC.
P.O. Box 379
North Hollywood, CA 91603

db/LIB is a trademark of AJS Publishing, Inc.
Microsoft is a trademark of Microsoft Corp.
dBASE III is a registered trademark of Ashton-Tate

"Professionals in Software"

YES!

Now you can put all the performance, speed, and programming convenience of **Microsoft's New QuickBASIC** compiler to work on your dBASE files.

YES!

db/LIB creates and manages DBF, NDX, and DBT files that are byte-for-byte identical to dBASE III files.

YES!

Using **db/LIB**, programs written in **QuickBASIC** can read and write existing dBASE II and dBASE III+ files, without conversion.

YES!

db/LIB works with QuickBASIC 3.0, non-coprocessor version.

YES!

QuickBASIC and db/LIB open up additional processing options, like graphics and indexing on memo files. So powerful you can build partial indexes, keep more files open, and even write your own dBASE utilities!

QB+db/LIB=DBMS

The combination of db/LIB and QuickBASIC provides you a very flexible, alternative development environment for existing dBASE installations.

ORDER NOW!

db/LIB is available today, and can return your investment the first time you use it.

Listing Four *(Listing continued, text begins on page 140.)*

```
IMPLEMENTATION MODULE Complex;

FROM MathLib0 IMPORT sqrt, sin, cos;

TYPE

    ComplexRecord = RECORD
                       Reel,
                       Imag : REAL;
                    END;

    (* opaque type mus be a pointer *)
    Complex = POINTER TO ComplexRecord;

PROCEDURE RealImagComplex(Re, Im : REAL) : Complex;
(* Convert from Real/Imaginary numbers to opaque complex numbers *)

VAR C : Complex;

BEGIN
    NEW(C);
    C^.Reel := Re;
    C^.Imag := Im;
    RETURN(C)
END RealImagComplex;

FUNCTION PolarComplex(Angle, Modulus : REAL) : Complex;
(* Convert from polar coordinates to opaque complex numbers *)

VAR C : Complex;

BEGIN
    NEW(C);
    C^.Reel := Modulus * sin(Angle);
    C^.Imag := Modulus * cos(Angle);
    RETURN(C)
END PolarComplex;

PROCEDURE GetRealImag(MyComplex : Complex; VAR Re, Im : REAL
                                            (* output *));
(* Convert opaque complex numbers into Real/Imaginary components *)

BEGIN
    Re := MyComplex^.Reel;
    Im := MyComplex^.Imag;
END GetRealImag;


PROCEDURE GetPolar(MyComplex : Complex; VAR Angle, Modulus : REAL
                                                 (* output*));
(* Convert opaque complex numbers into polar components *)
BEGIN
    WITH MyComplex DO
      Modulus := sqrt(Reel*Reel + Imag*Imag);
      Angle   := Imag / Reel;
    END;
END GetPolar;

PROCEDURE AddComplex(C1, C2 : Complex) : Complex;

VAR C : Complex;
    Re, Im : REAL;
BEGIN
    (* Get first complex number *)
    Re := C1^.Reel;
    Im := C1^.Imag;
    (* Get second complex number *)
    Re := Re + C2^.Reel;
    Im := Im + C2^.Imag;
    (* Update result *)
    C^.Reel := Re;
    C^.Imag := Im;
    RETURN(C)
END AddComplex;

END Complex.
```

End Listings

Dbase*

programming tools
*Clipper, FoxBASE+, dBASE, QuickSilver

The UI Programmer

UI is the first professional code generator; we wrote UI for programmers who want to automate their work but cannot use code that is 'almost' good enough. If your user interfaces include bounce-bar menus, pop-up help screens and the other features of today's best programs, you will gain an order of magnitude in productivity with UI.

UI is a second generation, programmable product — so your code comes out your way. Application specific edits, for instance, can be placed in the UI 'template' which controls the generation. Edit the screen appearance until it 'looks and feels' perfect. Everytime you generate code, your special logic is preserved.

Speaking of editing the screen, UI includes a powerful, 3-D screen editor, so you can draw pop-up help boxes over your pull-down menus, over your application.

The Documentor

To run Doc, you just tell it the name of the main-line routine and make sure your printer has a lot of paper! (Sure, you can have the output go to the screen or a file, too.)

You can tailor your documentation to include any or all of: a table of contents, system tree diagram (main line is the root), hierarchy (box diagram) charts for each module, action diagrams (modern style flow charts) for each PRG or procedure, DBF listings (structure, indexes, more), where used/updated listings for fields and all variables — by module and by line number within each module.

Our written money-back satisfaction guarantee set a new standard when we began it in 1985. (Return rate to date: 0.6% and dropping!) No copy protection, royalties or other nonsense.

Suggested retail: $295 each, (800) support included. At your dealer today. Call us for a very special offer on our latest release! (800) 233-3569 or, in NY, (212) 406-7026.

WallSoft
The Computer Aided Software Engineering Corporation
233 Broadway, Suite 869, New York, NY 10279

CIRCLE 90 ON READER SERVICE CARD

YOUR SYSTEM'S KEY COMPONENT

The Only Magazine By And For Advanced Micro Users.

At last there is a magazine that brings you the strictly technical but practical information you need to stay up-to-date with the ever changing microcomputer technology . . . *Micro/Systems Journal. Micro/Systems Journal* is written with the needs of the systems integrator in mind—the individual who's involved in putting together the hardware and software pieces of the microcomputer puzzle.

In each issue of *Micro/Systems Journal* you'll find such useful and progressive articles as:

- Interfacing to Microsoft Windows
- Unix on the PC
- 80386 Programming
- High Resolution PC Graphics
- Using 80286 Protected Mode
- Multiprocessing and Multitasking

You'll get the hands-on, nuts and bolts information, insight and techniques that *Micro/Systems Journal* is famous for . . . in-depth tutorials, reviews, hints . . . the latest information on computer integration, networks and multi-tasking, languages, and operating systems . . . hard-hitting reviews.

To start your subscription to *Micro/Systems Journal*, simply fill out one of the attached cards or write to *Micro/Systems Journal*, 501 Galveston Dr., Redwood City, CA 94063. You'll receive a full year (6 issues) of *Micro/Systems Journal* for just $20, and enjoy the convenience of having M/SJ delivered to your doorstep each month. Don't wait . . . subscribe today!

MICRO/ SYSTEMS JOURNAL

FOR THE ADVANCED COMPUTER USER

SUBSCRIBE NOW AND SAVE OVER 15% OFF THE NEWSSTAND PRICE!

15% SAVINGS

Yes! I want to subscribe to Micro/Systems Journal™ and save over 15% off the cover price.

☐ 1 Year (6 issues) $20 ☐ 2 Years (12 issues) $35

☐ Please charge my: ☐ Visa ☐ MasterCard ☐ American Express

☐ Payment enclosed ☐ Bill me later

Card # ______ Exp. date ______

Signature ______

Name ______

Address ______

City ______ State ______ Zip ______

Savings based on a full one-year cover price of $23.70. Canada and Mexico add $3 for surface mail, $7 for airmail per year. All countries add $12 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc.

3021

15% SAVINGS

Yes! I want to subscribe to Micro/Systems Journal™ and save over 15% off the cover price.

☐ 1 Year (6 issues) $20 ☐ 2 Years (12 issues) $35

☐ Please charge my: ☐ Visa ☐ MasterCard ☐ American Express

☐ Payment enclosed ☐ Bill me later

Card # ______ Exp. date ______

Signature ______

Name ______

Address ______

City ______ State ______ Zip ______

Savings based on a full one-year cover price of $23.70. Canada and Mexico add $3 for surface mail, $7 for airmail per year. All countries add $12 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc.

3021

15% SAVINGS

Yes! I want to subscribe to Micro/Systems Journal™ and save over 15% off the cover price.

☐ 1 Year (6 issues) $20 ☐ 2 Years (12 issues) $35

☐ Please charge my: ☐ Visa ☐ MasterCard ☐ American Express

☐ Payment enclosed ☐ Bill me later

Card # ______ Exp. date ______

Signature ______

Name ______

Address ______

City ______ State ______ Zip ______

Savings based on a full one-year cover price of $23.70. Canada and Mexico add $3 for surface mail, $7 for airmail per year. All countries add $12 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc.

3021

No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

For the Advanced Computer User
Micro/Systems Journal

Box 3713
Escondido, CA 92025-9843

No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

For the Advanced Computer User
Micro/Systems Journal

Box 3713
Escondidio, CA 92025-9843

No Postage
Necessary
If Mailed
In The
United States

BUSINESS REPLY MAIL

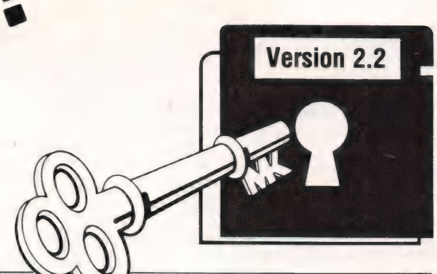
FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

For the Advanced Computer User
Micro/Systems Journal

Box 3713
Escondidio, CA 92025-9843

MASTER*KEY Unlocks Everything!



turn this into this!

```
C:>DEBUG PROGRAM.COM
-D100 136
8848:0100  EB 18 49 6E 63 6F 72 72-65 63 74 20 44 4F 53 20  k.Incorrect DOS
8848:0110  76 65 72 73 69 6F 6E 0D-0A 24 50 B4 30 CD 21 86  version..$P40M!.
8848:0120  E0 3D 36 01 72 05 3D 0A-02 76 09 BA 02 01 B4 09  `=6.r.=..v.:..4.
8848:0130  CD 21 CD 20 58 EB 2F                             M!M Xk/
-Q
```

```
H00100: JMP     Short H0011A                               ;00100 EB18           --
;-----------------------------------------------------------
        DB      "Incorrect DOS version"                    ;00102 496E636F727265
        DB      0Dh                                        ;00117
        DB      0Ah                                        ;00118
        DB      "$"                                        ;00119 24
;-----------------------------------------------------------
H0011A: PUSH    AX                                         ;0011A 50             P
        MOV     AH,30h                                     ;0011B B430           _0
        INT     21h               ;1-DOS_Ver_Number        ;0011D CD21           _!
        XCHG    AH,AL                                      ;0011F 86E0           __
        CMP     AX,0136h                                   ;00121 3D3601         =6_
        JB      H0012B            ; . . . . . . . . . .    ;00124 7205           r_
        CMP     AX,020Ah                                   ;00126 3D0A02         =__
        JBE     H00134            ; . . . . . . . . . .    ;00129 7609           v_
H0012B: MOV     DX,0102h                                   ;0012B BA0201         ---
        MOV     AH,09h                                     ;0012E B409           __
        INT     21h               ;1-Display_String        ;00130 CD21           _!
        INT     20h               ;TERM_normally:20h       ;00132 CD20           -
;-----------------------------------------------------------
H00134: POP     AX                                         ;00134 58             X
        JMP     Short H00166                               ;00135 EB2F           _/
;-----------------------------------------------------------


MASTER*KEY XREF - PROGRAM.XRF                                           Page 1

0102h                :    121   2F5   301   320
020Ah                :    126
03CBh                :    12B
1-Display_String     :    130   591   610
1-DOS_Ver_Number     :    11D
H00100               :    100
H0011A               :    100   11A
H0012B               :    124   12B
H00134               :    129   134
H00166               :    135
TERM_normally:20h    :    132
```

NOTE: The cross-reference is by memory location within the program file!

NOTE: The output is totally Microsoft MASM-compatible.

(not copy protected)

MASTER*KEY
No Other Product Comes Close!

An EXPERT may not know the solution, but always knows where to find it.

MASTER*KEY HELPS ANYONE solve those confusing and frustrating software puzzles more rapidly and easily than any other software available, at any cost! It gives you know-how within hours that may otherwise take years of experience. Create a new program from an old one. DON'T REINVENT THE WHEEL!

MASTER*KEY - Smart!
MASTER*KEY is an intelligent self-documenting MS-DOS reverse assembler. Its sophisticated procedures swiftly race through massive and baffling object code files to effortlessly discover potential trouble spots.

MASTER*KEY - Educational!
YOU DON'T NEED TO KNOW ASSEMBLY LANGUAGE! MASTER*KEY will take any program from your IBM-compatible computer and return fully-documented, easily-understood assembly language source code (Microsoft MASM 4.0 compatible).

MASTER*KEY - Easy To Use!
MASTER*KEY works both automatically from the DOS command line or interactively from menus similar to Lotus Corporation's 1-2-3 or Symphony. No need to remember any new commands or continually refer to a manual. Use it immediately!

Minimum System Requirements:
256K + 8088/8086/80186/80286/80386 PC
MS-DOS or PC-DOS 2.0 +
One 360K DSDD Floppy Drive (IBM PC Format)

MS-DOS is a trademark of Microsoft.
PC-DOS is a trademark of IBM.

MASTER*KEY will guide you step by step to:

1. Help you learn assembly language, if you desire.
2. Discover how any program runs or why it doesn't.
3. Alter or remove unwanted object code from any program.
4. Incorporate routines from compiled programs into other assembly language, Basic, C, or Pascal programs.
5. Make software more compatible with your computer. Be certain a questionable program won't damage your system BEFORE you run it.
6. Modify software to operate with other versions of DOS.
7. Customize your COMMAND.COM or other executable program directly or by reassembling your altered MASTER*KEY source code.

Order Now! Just $79^{95}

Phone orders accepted on MC or VISA
Send checks to:
Sharpe Systems Corporation
2320 E Street, La Verne, CA 91750

$82.45 (includes $2.50 shipping)
$87.65 in California (includes tax & shipping)
C.O.D. orders add $2.00

(714) 596-0070

Dealer/Distributor Inquiries Welcome

Sharpe Systems Corporation
2320 E STREET, LA VERNE, CA 91750   714-596-0070

CIRCLE 176 ON READER SERVICE CARD

MASTER*KEY should not be confused with any public domain or share ware software that may have a similar name or be a similar product.

"The Ada programming language shall be the single, common, high order programming language for...

"...all computers that are integral to, physically a part of, dedicated to, or essential in real time to a performance of the mission of weapon systems... used for specialized training, diagnostic testing and maintenance, simulation, or calibration of weapon systems... used for research and development of weapon systems...Use of validated compilers is required...this directive is effective immediately."
—DoD Directive 3405.2, 3/30/87.

"...Defense computer resources used in intelligence systems, for the command and control of military forces...all major software upgrades...all other applications (some exceptions) in keeping with the long range goal of establishing Ada as the primary DoD higher order language...waivers to the policy...shall be strictly controlled and closely reviewed... this directive is effective immediately."
—DoD Directive 3405.1, 4/2/87.

AdaNow

CIRCLE 273 ON READER SERVICE CARD

Special Introductory Offer For Apollo, H-P, and Sun Users

AdaNow

New Alsys Toolset For 68000 Ada Builds Unique Project Environment

Organizations serious about the 680X0 architecture, and serious about working with the government, want a lot more than just validated Ada compilers. They want quality solutions; production quality compilers and quality programming tools.

Just what Alsys offers. Alsys' new 68000 Ada Developer's Toolset includes:

- **AdaProbe**, a unique source-level symbolic debugger and program viewer;
- **AdaXref**, an inter-unit cross-referencing utility;
- **AdaReformat**, a pretty printing tool for reformatting source files to selectable conventions; and
- **AdaMake**, an automatic recompilation facility.

Consider, too, all those special Ada "manager tools" that are part of the Alsys Version 3 compilation system: the **Family Manager**, the **Unit Manager**, and the **Library Manager**.

Together, they implement the new Alsys **Multi-Library Environment** that allows teams of programmers to share thousands of logically organized compilation units.

Alsys 68000 compilers are in a class by themselves; highest code quality, maturity, reliability, robustness, superior optimization technology, unexcelled error messages...And now, with the new development tools, they are at the core of an Ada project environment unique in the industry.

Here is our special INTRODUCTORY OFFER. Between now and October 31, 1987, order any of our 68000 Ada compilers and we will include the complete Toolset FREE. AdaProbe, AdaXref, AdaReformat, AdaMake.

Ada is NOW. Alsys solutions are NOW. Call or Write.

alsys

Alsys, Inc. • 1432 Main Street • Waltham, MA 02154 • U.S.A. • Phone: (617) 890-0030
Offer valid in U.S.A. and Canada, only.

The Many Facets of
Quality
alsys

_____ YES, I'm interested in your Introductory Offer. Send more information on the Toolset and your 68000 compilers.

_____ Send me your free brochure, *The Many Facets of Quality.*

Name_____

Company_____

Address_____

City_____ State_____ Zip_____

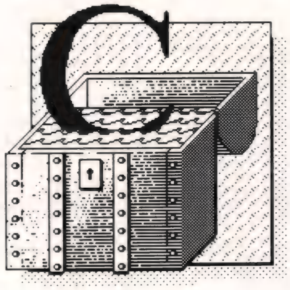
Phone_____

Alsys, Inc. • 1432 Main Street • Waltham, MA 02154

CIRCLE 273 ON READER SERVICE CARD

*Ada is a registered trademark of the U.S. Government (AJPO).

Language Wars Over Cs



There seems to be a plethora of new C compilers hitting the market at present. This month I'm going to look at four of them: Datalight's Optimum-C; Microsoft's Quick C; Microsoft's C compiler, Version 5.0; and Borland's Turbo C.

I'm going to take a different approach from most reviews and leave out the traditional tables and benchmarks. I'm doing this for several reasons. First, tables of features don't usually contain new information; you can read the advertisements as well as I can, and there's no point in my duplicating that information here. Next, I've found that benchmarks aren't a particularly reliable way to evaluate compilers. That is, a compiler that does poorly in compiling a single benchmark subroutine often does well when you use it for a large, complex program and vice versa. Also, raw speed is not the most important thing in a compiler, at least not to me.

As a programmer I spend most of my day trying to debug computer programs. Consequently, a good development environment is critical to me. I frankly don't care if my program takes 50 milliseconds longer to run or is 512 bytes larger than it could be, provided that I can minimize the time I spend in debugging and the

by Allen Holub

code quality is adequate. Most programs are I/O-bound anyway. That is, the fastest execution speed in the world doesn't speed up the rate at which you can update the screen or get input from a human being.

Of course, my own criteria may be different from yours. If you're doing engineering applications where you measure run time in hours rather than seconds, a few milliseconds here and there can be significant. If you fall into that category, the following reviews will still be useful but you should probably look at a review that has benchmarks, too.

All four compilers are quite good from the language perspective. They all support the complete C language (including bit fields), and they all incorporate various ANSI extensions, such as function prototypes and structure assignment. They all support floating point, including in-line 8087 code. They all include a version of the Unix make utility. They all come with the start-up module source code and have a good set of DOS interface functions. There are differences, however.

Optimum-C, Version 3.0

Datalight's Optimum-C is a pleasant surprise. First of all, it generates spectacularly good code—better than any of the other compilers reviewed here. It not only does the usual optimizations—such as constant folding, strength reduction, branch optimization, and aliasing—but it does some very sophisticated stuff too—constant and variable propagation, dead assignment elimination, automatic register allocation, global common subexpression elimination, loop invariant removal, and loop induction variables. The four basic memory models are supported (no huge or tiny), but mixed-model programming is difficult to do.

Optimum-C has good ROM support, though it doesn't come with the special linker that you need to locate specific segments in particular places in memory. (One of these days I'll write a linker for C Chest.) It provides an integer-only option that lets you compile without any floating-point support, and it provides you with a stripped-down start-up module that helps put together a ROMed environment.

Datalight includes a list of known bugs on the distribution disks. All compilers have bugs, but Datalight is honest enough to admit it, thereby helping you program around them. This honesty should be commended. It's a small thing, but it helps a lot when you want to get your program running.

The two drawbacks of Optimum-C are the lack of debugging support and a library that is too small. You pretty much have to rely on embedded *printf( )* statements to debug your programs. For this reason I don't really recommend the compiler if you're not already an experienced C programmer. The run-time library is adequate but just so. All the essential functions are there, but there's no gravy—for example, there's a *getenv( )* but no *putenv( )*. The library does include some useful stuff not found in the other compilers. There's a mouse-interface library and a set of interrupt-management functions that are useful for terminate-and-stay-resident utilities (Microsoft C has these too, however).

The compiler includes all the li-

10 Important Reasons C Programmers Use Our File Manager

1. It's written in C.
Clearly the growing language of choice for applications that are fast, portable and efficient. All of db_VISTA's source code is written in C.

2. It's fast – almost 3 times faster than a leading competitor.
Fast access that comes from the unique combination of the B-tree indexing method and the "network" or direct "set" relationships between records. A winning combination for fast performance.

3. It's flexible.
Because of db_VISTA's combination of access methods, you can program to your application needs with ultimate design flexibility. Use db_VISTA as an ISAM file manager or to design database applications. You decide how to optimize run-time performance. No other tool gives you this flexibility without sacrificing performance.
db_VISTA is also well behaved to work with most any other C libraries!

4. It's portable.
db_VISTA operates on most popular computers and operating systems like UNIX, MS-DOS and VMS. You can write applications for micros, minis, or even mainframes.

5. Complete Source Code available.
We make our entire C Source Code available so you can optimize performance or port to new environments yourself.

6. It uses space efficiently.
db_VISTA lets you precisely define relationships to minimize redundant data. It is non-RAM resident; only those functions necessary for operation become part of the run-time program.

7. Royalty free run-time.
Whether you're developing applications for yourself or for thousands, you pay for db_VISTA or db_QUERY only once. If you currently pay royalties to someone else for your hard work, isn't it time you switched to royalty-free db_VISTA?

db_VISTA™

Features
- **Multi-user** support allows flexibility to run on local area networks
- **File structure** is based on the B-tree indexing method
- **Transaction processing** assures multi-user consistency
- **File locking** support provides read and write locks
- **SQL-based db_QUERY** is linkable
- **File transfer** utilities included for ASCII, dBASE optional
- **Royalty-free** run-time distribution
- **Source Code** available
- **Data Definition Language** for specifying the content and organization of your files
- **Interactive database access** utility
- **Database consistency check** utility

File Management Record and File Sizes
- Maximum record length limited only by accessible RAM
- Maximum records per file is 16,777,215
- Maximum file size limited only by available disk storage
- Maximum of 256 index and data files
- Key length maximum 246 bytes
- No limit on number of key fields per record
- No limit on maximum number of fields per record

Operating System & Compiler Support
- **Operating systems:** MS-DOS, PC-DOS, UNIX, XENIX, UNOS, ULTRIX, Microport, VMS
- **C compilers:** Lattice, Microsoft, IBM, DeSmet, Aztec, Computer Innovations, Turbo C, XENIX and UNIX

8. SQL-based db_QUERY™
Add our new C-linkable, SQL-based, ad hoc query and report-writing companion product to provide a simple relational view of your db_VISTA applications. Without compromising speed.

9. Free tech support.
60 days of free technical and application development support for every Raima product. Of course, extended support and training classes are also available at your place or ours.

10. Upward database compatibility
Start out with file management in a single-user PC environment—then move up to a multi-user LAN or a VAX database application with millions of records. You'll still be using db_VISTA. That's why so many C programmers are choosing db_VISTA.

But don't just take our word for it.
"Raima's customer support and documentation are excellent. Source code availability and royalty-free run-time is a big plus."
Dave Schmitt, President
Lattice, Inc.

"db_VISTA has proved to be an all-round high performer in terms of fast execution, flexibility and portability, and has undoubtedly saved us much time and development effort."
John Adelus, Hewlett-Packard
Office Productivity Division

30-day Money Back Guarantee!
Try db_VISTA in your environment for 30 days and prove it to yourself. If not completely satisfied, return it for a full refund.

Price Schedule

	db_VISTA	db_QUERY
☐ Single user	$ 195	$ 195
☐ Single user w/Source	$ 495	$ 495
☐ Multi-user	$ 495	$ 495
☐ Multi-user w/Source	$ 990	$ 990
NEW:		
☐ VAX Multi-user	$ 990	$ 990
☐ VAX Multi-user w/Source	$1980	$1980

Order Now.
Put db_VISTA to work in your application program. Ordering is easy—simply call toll-free. We'll answer your technical questions and get you started. Call today.

Call Toll-Free Today!
1 (800) db-RAIMA
(800/327-2462) or
206/828-4636

RAIMA™ CORPORATION

3055 - 112th NE, Bellevue, WA 98004 USA
(206) 828-4636 Telex: 6503018237 MCI UW

DD107

(continued from page 124)

brary sources and an adequate full-screen editor that might be useful if you don't have one already.

Quick C and Microsoft C, Version 5

I should preface my comments by saying that I'm looking at beta versions of both Quick C and the C compiler. Consequently, some of the things I'm saying here may become untrue later. (Both programs should be released in the final version by the time this article makes it into print.) I'll report back periodically as I get updates of these products. I'm also willing to give the people at Microsoft the benefit of the doubt, for the present. If they say something is going to be fixed, I'll believe them. Nonetheless, if the final version of the product still has problems, I'll discuss the problems in depth in this column.

The similarities between Quick C and Turbo C are unavoidable. They both include a development environment that incorporates an editor, compiler, and so forth. Neither of the editors is anything to write home about; both are adequate for fixing occasional errors that might pop up in debugging. Neither is really adequate for writing an actual program however, so you will probably do your initial typing outside the Quick C or Turbo C environment. In fact, these much-touted user interfaces are pretty worthless for the most part. I'd as soon use my own editor and compile with the command-line versions of the compilers. I see little point in wading through infinite menus when I can do the same job faster from the command line. How long can it take to learn a few command-line switches anyway? Fortunately, both Turbo and Quick C provide an easy-to-use, command-line version of the compiler.

There's a certain amount of overlap between Quick C and the full, Version 5, compiler, too. In fact, Quick C is included in the full compiler package. The main differences are code quality and compile time. The full compiler is slower, but it generates much better, heavily optimized code. Quick C, on the other hand, is lightning fast but the output code isn't great. There's also a new, much faster version of LINK provided with the package.

Both Quick C and the full compiler use the same libraries, literally—they use the same .lib files on the disk. This is a real boon if you're doing serious development work—you can use Quick C to develop individual modules, and once they're debugged, you can recompile with the full compiler to get better code quality.

Quick C provides a phenomenally good development environment. It has about three-quarters of CodeView built into it, so it not only finds syntax errors for you but it also helps you debug. You can use a command-line version of Quick C to generate CodeView-compatible files, however, so you can have both fast compile time and the full CodeView if you need it.

For those of you who aren't familiar with the CodeView debugger, it is a source-level debugger that really works, and it has become an essential part of my development environment. You can actually watch program execution at the source level; you can even see local and global variables change as they are modified. You can set breakpoints on variables being modified, on expressions becoming true, even on ranges of memory being modified. I discussed the debugger in depth in the November 1986 C Chest. My program development time has decreased by at least 25 percent since I started using this debugger. I can't live without it. Quick C is essentially CodeView with a compile instruction.

Quick C handles multiple-module files quite easily. You create a program list that has an entry for each file or library in the program. Quick C uses this list to create a makefile automatically and then assembles the program in a manner similar to the way make does (Turbo C uses a similar mechanism).

Quick C is really an incremental compiler not an interpreter. It compiles the entire module at lightning speed—comparable in all respects to Turbo C—without stopping. It has a "go to next error" function key in the editor that lets you find errors painlessly. It positions you at the line that contains the error, then prints the error message in a window at the bottom of the screen. A context-sensitive help feature gives you an on-line reference to all the library routines, showing you a function prototype and capsule description of any library function that is highlighted by the cursor.

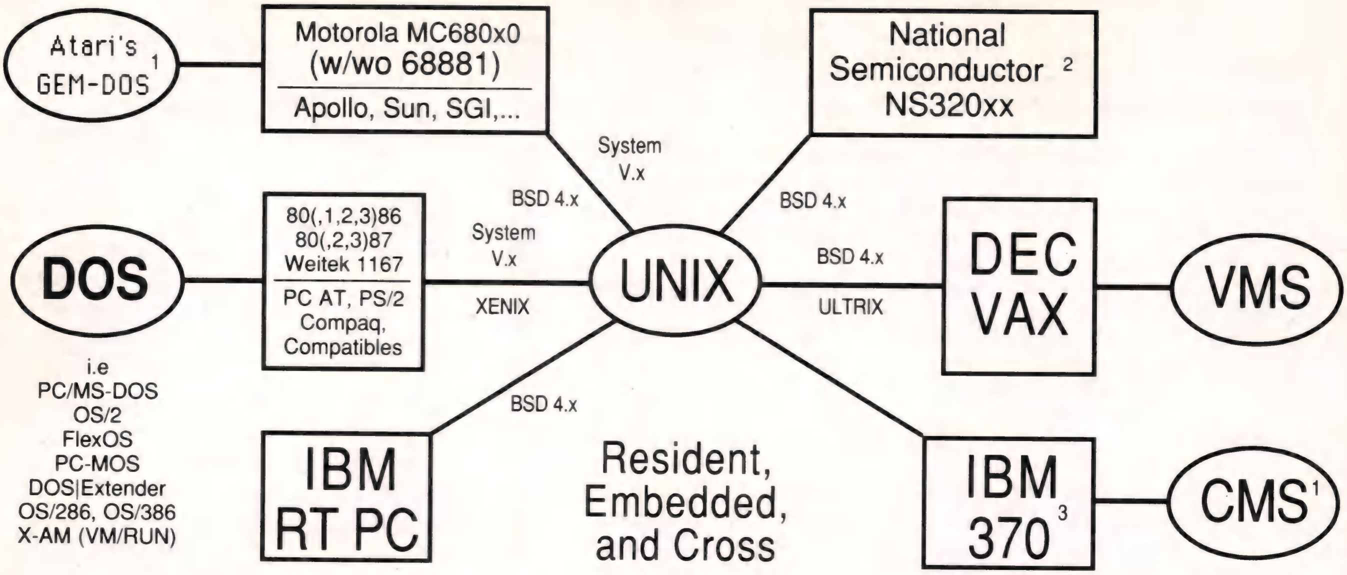
The Microsoft run-time library, used by both Quick C and the full compiler, is the best out of those reviewed here. It is very Unix compatible and is packed with useful routines, including a set of graphics functions (new to this version) and very good support for terminate-and-stay-resident utilities (also new). The graphics functions let you do the basic stuff (lines, circles, ellipses, and area fill) and supports all the IBM video adapters (but, unfortunately, not the Hercules graphics card). The library documentation is among the best I've seen—better than that of any of the other compilers I'm reviewing here. It has one page per function, with the function name across the top of the page. The functions are listed in strict alphabetical order so that they're easy to find, and most entries have an example of how to use them.

Microsoft is finally selling the library source code too—useful if you're either porting to another environment or need to correct bugs that are bound to show up in the new routines. Microsoft is also accelerating the release schedule from yearly to semiannually so that you can get a compiler update with fixed bugs every six months instead of having to wait a year. This last change makes the library sources a little less important, but it's still good that they're being released.

Turbo C

Last, and least, on the list is Borland's Turbo C. I finally got my copy of Turbo C in the mail, several months after Borland had not only started advertising it but also collecting money for it. The cover letter started ominously with: ". . . right on schedule, here's Turbo C." In spite of the hype, however, and in spite of the seeming popularity of the product (at least judging by the sales figures), Turbo C comes in last when compared to the compilers I've just discussed. It's not so much that Turbo C isn't a good product but

C and Pascal on:



We cut our teeth on UNIX, but have become famous on MS-DOS, which we enhanced with our UNIX-like **DOS Helper** ™ utilities: find (including tar), fgrep, cat, ls, mv, tail, uniq, and wc; and our superior optimizing compilers: **Professional Pascal** ™ and **High C** ™ on the PC are now well-respected by organizations such as Ansa, Ashton-Tate, AutoDesk, Boeing (BCS), Daisy Systems Corp., Deloitte Haskins & Sells, Digital Research, GE, IBM, Lifetree, Migent, Multimate, NYU, Silvar-Lisco, Sky Computers, Symantec, Xerox/Ventura, ...and *Computer Language* magazine; *Dr. Dobbs' Journal; PC Tech Journal; PC Magazine;* and the *Journal of Pascal, Ada, and Modula-2.*

We supplied the **first** compilers generating **32-bit protected-mode code for the 80386** under MS-DOS (since 11/86). And our newly upgraded MS-DOS real-mode compilers were used by Symantec for their Q&A™ product to exploit the power of the 80386 real-mode instruction set. (**HC v1.4 and PP v2.7 released May 1987.**)

Our **C Validation Suite** will blow your C compiler out of the sea, while our C compiler tracks the emerging ANSI Standard and generates tighter code with far better lint-like feedback help than competing compilers.

And you'll love **Professional Pascal**'s Ada-like packages, true data abstraction, C-like bit manipulation, and much more, along with the tight code that is linkable with **High C**, or other C, object modules (and vice versa).

Our **Translator Writing System (TWS)** goes far beyond LEX and YACC, with fully automatic error recovery...

All uniformly implemented on UNIX, VMS, CMS, MS-DOS, FlexOS, ...

***Professional developers** in need of **industrial-strength tools** contact:*

Meta Ware™
INCORPORATED

MetaWare Incorporated
903 Pacific Avenue, Suite 201
Santa Cruz, CA 95060-4429

(408) 429-6382

Telex: 493-0879 (META UI)

PC Tech Journal's conclusion:

The Clear Choice for Large Programming Projects.

Name ______	CL 05/87	*Circle what interests you:*
Company ______	Product:	PP HC TWS Helper (DOS only)
Address ______	Platform:	V.x 4.x DOS FlexOS VMS CMS
City, ST ______ Zip ______		Sun Apollo Atari VAX 370
Phone (___) ______		8086-family 80386 680x0 32032

© 1987 MetaWare Incorporated. MetaWare, High C, Professional Pascal, and DOS Helper are trademarks of MetaWare Incorporated. Others/ owners: Ada/DoD; Apollo/Apollo; Atari/Atari; DEC,VAX,VMS/DEC; FlexOS,GEM-DOS/DRI; IBM,RT PC/IBM; MS-DOS/Microsoft; Q&A/Symantec; Sun/Sun Microsystems; UNIX/AT&T.

Footnotes: 1. Atari, CMS versions available 10/87. 2. NS320xx version by special order. 3. UNIX not yet available on 370.

CIRCLE 95 ON READER SERVICE CARD

NEW VERSION

PC/VI™

UNIX's VI Editor Now Available For Your PC!

Are you being as productive as you can be with your computer? An editor should be a tool, not an obstacle to getting the job done. Increase your productivity today by choosing **PC/VI**—a COMPLETE implementation of UNIX* VI version 3.9 (as provided with System V Release 2).

PC/VI is an implementation of the most powerful and most widely used full-screen editor available under the UNIX operating system. The following is only a hint of the power behind **PC/VI:**

- Global search or search and replace using regular expressions
- Full undo capability
- Deletions, changes and cursor positioning on character, word, line, sentence, paragraph, section or global basis
- Editing of files larger than available memory
- Shell escapes to DOS
- Copying and moving text
- Macros and Word abbreviations
- Many controllable options including Auto-indent, Showmatch, and Wrap Margin
- Filter text through external programs AND MORE!

Don't take it from us. Here's what some of our customers say: "Just what I was looking for!", "It's great!", "Just like the real VI!". "The documentation is so good I have already learned things about VI that I never knew before." — *IEEE Software*, September 1986.

PC/VI is available for IBM-PC's and generic MS-DOS† systems for only $149. Included are CTAGS and SPLIT utilities, TERMCAP function library, and an IBM-PC specific version which enhances performance by as much as TEN FOLD!

PC/TOOLS™

What makes UNIX so powerful? Sleek, Fast, and **POWERFUL** utilities! UNIX gives the user not dozens, but hundreds of tools. Now the most powerful and popular of these are available for your PC! Each is a complete implementation of the UNIX program. Open up our toolbox and find:

• ASA	• COMM	• DIFF3	• MV	• SEE	• TR
• BANNER	• CP	• FIND	• OD	• SORT	• TOUCH
• BFS	• CUT	• GREP	• PASTE	• SPLIT	• UNIQ
• CAL	• DATE	• HEAD	• PR	• STRINGS	• WC
• CAT	• DIFF	• LS	• RM	• SUM	
• CHMOD	• DIFFH	• MAKE	• SED	• TAIL	

All of these for a limited introductory price of only $49.00; naturally, extensive documentation is included!

PC/SPELL™

Why settle for a spelling checker which can only compare words against its limited dictionary database when **PC/SPELL** is now available? **PC/SPELL** is a complete implementation of the UNIX spelling checker, renowned for its understanding of the rules of English! **PC/SPELL** determines if a word is correctly spelled by not only checking its database, but also by testing such transformations as pluralization and the addition and deletion of prefixes and suffixes. For only $49.00, **PC/SPELL** is the first and last spelling checker you will ever need!

Buy **PC/VI** and **PC/TOOLS** *now* and get **PC/SPELL** for only $1.00! Site licenses are available. Dealer inquiries invited. MA residents add 5% sales tax. AMEX, MC and Visa accepted without surcharge. Thirty day money back guarantee if not satisfied! Available in 5¼", 3½" and 8" disk formats. For more information call today!

*UNIX is a trademark of AT&T. †MS-DOS is a trademark of Microsoft.

UNIX TOOLS FOR YOUR PC

UNIX TOOLS FOR YOUR PC

CUSTOM SOFTWARE SYSTEMS
P.O. BOX 678 • NATICK, MA 01760
617•653•2555

CIRCLE 268 ON READER SERVICE CARD

C CHEST
(continued from page 126)

rather that the others are better.

Like Quick C, there are two interfaces to the compiler itself—a normal command-line interface and "the standard Borland integrated environment." The integrated environment is pretty worthless. Were I a Turbo Pascal user, I might like it better, but I'm not—the integrated environment does nothing but put extra steps between me and the compiled program. That is, it forces me to use an editor that I don't like; it forces me to wade through infinite menus to get anything done; and its menu system manages to make even simple things complex by adding too many steps to any process. Borland has made the classic mistake of confusing "user friendly" with "coddle the novice." The interface is only "friendly" until you know what you're doing.

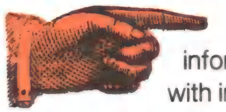
The biggest problem with the integrated environment is the complete lack of debugging support. The product does find syntax errors for you, in a manner similar to Turbo Pascal. It puts you into the editor at the error point so that you can change things and finish the compilations. Syntax errors, however, are the least of it. It takes me a few minutes to get the syntax errors out of a program, but it can take days to get the program debugged, and Turbo C gives you absolutely no help with real debugging. It has nothing like the CodeView-like environment provided in Quick C.

So, once you get rid of the excess baggage, what you have is an inexpensive and reasonably good C compiler. The code quality is better than Microsoft C, Version 4, but not as good as Version 5. It supports all six memory models and lets you do mixed-model programming. It's easy to use from the command line and generates reasonably good code. Rumor has it that it's the Wizard C compiler, and looking at the generated code, I have no reason to doubt this supposition. The compiler supports in-line assembly language and has a reasonably good library, the sources for which are available if you need them. The DOS support is good but not portable, and there are no graphics functions.

The library has a few problems,

For Free Info ...

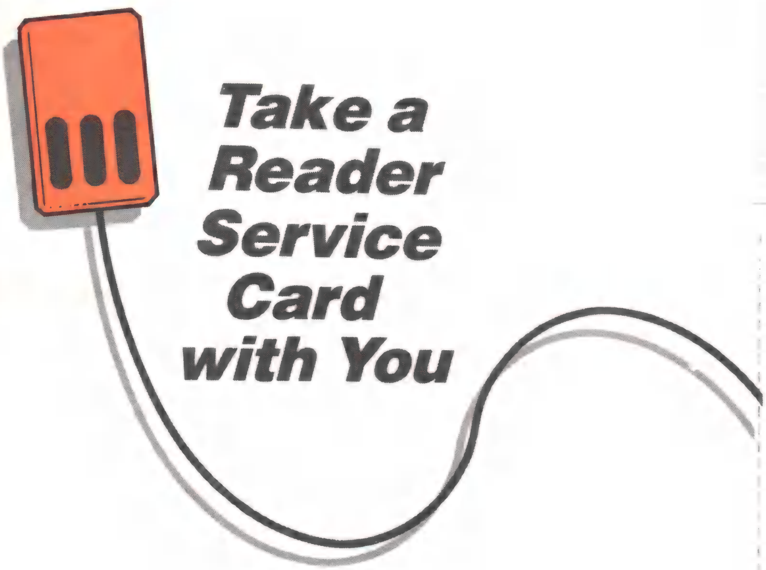
Start Here



Smart buyers start with *DDJ*'s free information card, a shopping center filled with information about the products and services advertised in this very issue: everything from software and systems to peripherals and professional support services.

And smart buyers can use this free information card to quickly and easily gather a comprehensive file of facts, figures and product specs to sort out competing claims. Using *DDJ*'s free information card can prevent you from making the wrong, costly buying decision.

Be a smart shopper. Complete and mail this postage paid card today!



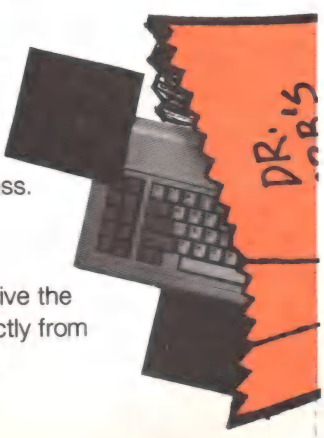
Take a Reader Service Card with You

It's Easy as ...

1. Circle the appropriate free information numbers, referring to the advertiser index for more information.

2. Fill in your name and address.

3. Mail today—postage is absolutely free. You'll receive the product information you need directly from the manufacturers, thanks to *DDJ*.



NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT #217, CLINTON, IOWA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of Software Tools
FOR THE PROFESSIONAL PROGRAMMER

P.O. Box 2157
Clinton, Iowa 52735-2157

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT #217, CLINTON, IOWA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of Software Tools
FOR THE PROFESSIONAL PROGRAMMER

P.O. Box 2157
Clinton, Iowa 52735-2157

Dr. Dobb's Journal of

Software Tools

FOR THE PROFESSIONAL PROGRAMMER

Use this card for FREE, FAST information about the products and services listed in this issue. Simply circle the appropriate numbers below.

1	26	51	76	101	126	151	176	201	226	251	276	301	326	351	376
2	27	52	77	102	127	152	177	202	227	252	277	302	327	352	377
3	28	53	78	103	128	153	178	203	228	253	278	303	328	353	378
4	29	54	79	104	129	154	179	204	229	254	279	304	329	354	379
5	30	55	80	105	130	155	180	205	230	255	280	305	330	355	380
6	31	56	81	106	131	156	181	206	231	256	281	306	331	356	381
7	32	57	82	107	132	157	182	207	232	257	282	307	332	357	382
8	33	58	83	108	133	158	183	208	233	258	283	308	333	358	383
9	34	59	84	109	134	159	184	209	234	259	284	309	334	359	384
10	35	60	85	110	135	160	185	210	235	260	285	310	335	360	385
11	36	61	86	111	136	161	186	211	236	261	286	311	336	361	386
12	37	62	87	112	137	162	187	212	237	262	287	312	337	362	387
13	38	63	88	113	138	163	188	213	238	263	288	313	338	363	388
14	39	64	89	114	139	164	189	214	239	264	289	314	339	364	389
15	40	65	90	115	140	165	190	215	240	265	290	315	340	365	390
16	41	66	91	116	141	166	191	216	241	266	291	316	341	366	391
17	42	67	92	117	142	167	192	217	242	267	292	317	342	367	392
18	43	68	93	118	143	168	193	218	243	268	293	318	343	368	393
19	44	69	94	119	144	169	194	219	244	269	294	319	344	369	394
20	45	70	95	120	145	170	195	220	245	270	295	320	345	370	395
21	46	71	96	121	146	171	196	221	246	271	296	321	346	371	396
22	47	72	97	122	147	172	197	222	247	272	297	322	347	372	397
23	48	73	98	123	148	173	198	223	248	273	298	323	348	373	398
24	49	74	99	124	149	174	199	224	249	274	299	324	349	374	399
25	50	75	100	125	150	175	200	225	250	275	300	325	350	375	400

October '87: Use before January 31, 1988

Name ______________________________

Title ______________________________

Company ____________________ Phone __________

Address ______________________________

City/State/Zip ______________________________

Dr. Dobb's Journal of

Software Tools

FOR THE PROFESSIONAL PROGRAMMER

Use this card for FREE, FAST information about the products and services listed in this issue. Simply circle the appropriate numbers below.

1	26	51	76	101	126	151	176	201	226	251	276	301	326	351	376
2	27	52	77	102	127	152	177	202	227	252	277	302	327	352	377
3	28	53	78	103	128	153	178	203	228	253	278	303	328	353	378
4	29	54	79	104	129	154	179	204	229	254	279	304	329	354	379
5	30	55	80	105	130	155	180	205	230	255	280	305	330	355	380
6	31	56	81	106	131	156	181	206	231	256	281	306	331	356	381
7	32	57	82	107	132	157	182	207	232	257	282	307	332	357	382
8	33	58	83	108	133	158	183	208	233	258	283	308	333	358	383
9	34	59	84	109	134	159	184	209	234	259	284	309	334	359	384
10	35	60	85	110	135	160	185	210	235	260	285	310	335	360	385
11	36	61	86	111	136	161	186	211	236	261	286	311	336	361	386
12	37	62	87	112	137	162	187	212	237	262	287	312	337	362	387
13	38	63	88	113	138	163	188	213	238	263	288	313	338	363	388
14	39	64	89	114	139	164	189	214	239	264	289	314	339	364	389
15	40	65	90	115	140	165	190	215	240	265	290	315	340	365	390
16	41	66	91	116	141	166	191	216	241	266	291	316	341	366	391
17	42	67	92	117	142	167	192	217	242	267	292	317	342	367	392
18	43	68	93	118	143	168	193	218	243	268	293	318	343	368	393
19	44	69	94	119	144	169	194	219	244	269	294	319	344	369	394
20	45	70	95	120	145	170	195	220	245	270	295	320	345	370	395
21	46	71	96	121	146	171	196	221	246	271	296	321	346	371	396
22	47	72	97	122	147	172	197	222	247	272	297	322	347	372	397
23	48	73	98	123	148	173	198	223	248	273	298	323	348	373	398
24	49	74	99	124	149	174	199	224	249	274	299	324	349	374	399
25	50	75	100	125	150	175	200	225	250	275	300	325	350	375	400

October '87: Use before January 31, 1988

Name ______________________________

Title ______________________________

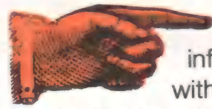
Company ____________________ Phone __________

Address ______________________________

City/State/Zip ______________________________

For Free Info ...

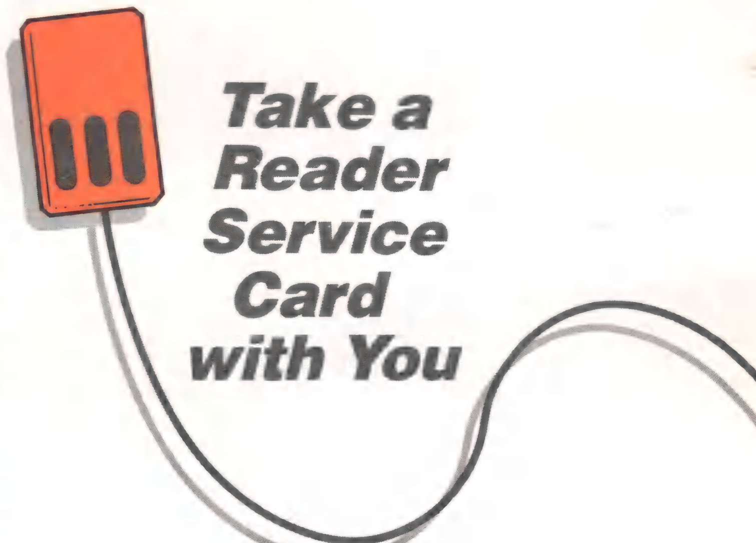
Start Here



Smart buyers start with *DDJ*'s free information card, a shopping center filled with information about the products and services advertised in this very issue: everything from software and systems to peripherals and professional support services.

And smart buyers can use this free information card to quickly and easily gather a comprehensive file of facts, figures and product specs to sort out competing claims. Using *DDJ*'s free information card can prevent you from making the wrong, costly buying decision.

Be a smart shopper. Complete and mail this postage paid card today!

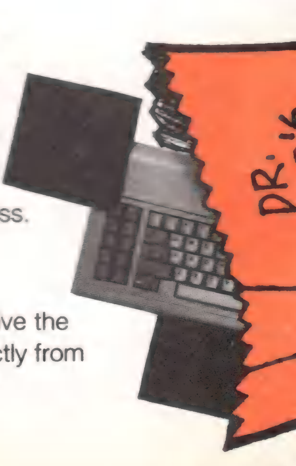


It's Easy as ...

1. Circle the appropriate free information numbers, referring to the advertiser index for more information.

2. Fill in your name and address.

3. Mail today—postage is absolutely free. You'll receive the product information you need directly from the manufacturers, thanks to *DDJ*.



The Advertiser Index

*This advertiser prefers to be contacted directly; see ad for phone number.

Advertising Sales Offices

Midwest
Charles Shively (415) 366-3600

Northern California/Northwest
Lisa Boudreau (415) 366-3600

Northeast
Cynthia Zuck (718) 499-9333
Martha Brandt (415) 366-3600

Southern California/AZ/NM/TX
Michael Wiener (415) 366-3600

Director of Marketing and Advertising
Ferris Ferdon (415) 366-3600

Telemarketing Representative/SE/SW USA
Cheri Blum (714) 761-0294

C CHEST
(continued from page 128)

however, most of which fall into the Unix-compatibility and documentation areas. For example, there's no Unix-compatible *signal( )* function, though there is a nonstandard mechanism to intercept the Ctrl-Break interrupt; there's a function called *ioctl( )* but this function is nothing like the Unix function with the same name; and so forth. These sorts of incompatibilities always mystify me. It's easy enough to do it right, so why don't they?

The library documentation is adequate but not nearly as good as Microsoft's. There's lots of nonstandard stuff with little or no explanation of how that stuff works. For example, the nonstandard *ioctl( )* subroutine evidently lets you change file attributes, but the documentation doesn't tell you how. I assume that the *DOS Technical Reference* would help, but Borland's documentation doesn't say one way or the other. In addition, examples of how to use library routines are few and far between, and the documentation assumes a lot in terms of previous knowledge needed to figure out the subroutine description. A thorough knowledge of DOS interfacing details is assumed throughout.

The manual's layout is poor. Borland has saved space by putting the documentation for several subroutines on a single page, thereby making things harder to find (there's no header with the subroutine names for that page across the top). More often than not, when you look up a subroutine, the entry refers you somewhere else to get the actual description. There are a few typos that cause problems too—for example, the *stime( )* function is described as follows: "*stime* returns a value of 0 is returned."

Conclusion

Quick C in the stand-alone version is better than Turbo C. It does everything that Turbo C does, and then some, incorporating very good debugging support that is totally absent from Turbo C (finding syntax errors alone is not sufficient). My Quick C is a beta version, so I can't really compare code quality, and in any event, all of the products are more or less on par. Nonetheless, Turbo C is, at present, between Quick C (which is a little poorer) and the full Microsoft compiler (which is considerably better). Microsoft says that the code quality will be considerably improved in Quick C's final release.

To my mind, the better debugging environment provided by Quick C far outweighs any code-quality considerations. If you're doing serious development, you'll use the full compiler anyway. Quick C's debugging environment is particularly useful if you're learning the language.

Datalight's Optimum-C gives the best code quality of these four compilers, and the library, though small, is adequate for most programs. It also provides the best support for ROMed code, and the library sources are included for free. The lack of debugging support is a serious omission, however, and I recommend it primarily to experienced programmers who are generating code to run outside the DOS environment or to programmers who need very efficient code and don't care about the small library. I do recommend it, though, in spite of these shortcomings. It's a shame that Datalight's C generates Lattice-compatible assembly language rather than Microsoft-compatible. Were this not the case, you could use Quick C to do your development and Optimum-C for the final compilation pass.

The complete Microsoft compiler package is the most powerful, but it's by far the most expensive, especially when you add in the cost of the library sources. If cost is not an issue, I think the Microsoft compiler (which includes Quick C) is the best bet. It combines a very good compiler with a complete (and Unix-compatible) library and a fantastic debugging environment. If cost is an issue, or if you're just learning the language, I'd go with Quick C alone. If you're doing serious production work, get the full compiler package.

Bug Report

There is a bug on page 95 of the June column listings. On line 215, change *&item* to *item*.

DDJ

Vote for your favorite feature/article.
Circle Reader Service **No. 6**.

'Faster and better in no time'

Create better, faster, higher quality and easier to read progams in a fraction of the time. Let your system do the work for you. With 23 powerful, state of the art tools for the IBM PC and compatibles, both beginners and experts will find programming a breeze.

- Powerful, common user interace
- Interactive and batch execution
- Online documentation
- Self-explanatory error messages

C Programmer's Toolbox
Volumes I & II

- Monitor program/system execution
- Beautify program listings
- Determine program flow/critical path
- Trace/verify variable usage
- And more....

At $79.95 per volume or $130 for both with a **30 day trial**, the Toolbox is simply the **best value** today. In addition, the documentation is packed with examples and tips on creating better programs.

Why waste time, **call or write us today.**

TM

MMC AD Systems
Box 360845 Milpitas, California 95035
(408) 263-0781

CIRCLE 192 ON READER SERVICE CARD

C CODE FOR THE PC

source code, of course

C Source Code

FSP (screen manager)	$400
Bar Code Generator (Codabar, 3 of 9, and other popular codes; price is per code)	$300
GraphiC 4.0 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$275
Vitamin C (MacWindows)	$200
Essential C Utility Library (400 useful C functions)	$160
Essential Communications Library (C functions for RS-232-based communication systems)	$160
Panache C Program Generator (screen-based database management programs)	$150
PC/IP (CMU/MIT TCP/IP implementation for PCs)	$100
B-Tree Library & ISAM Driver (file system utilities by Softfocus)	$100
The Profiler (program execution profile tool)	$100
QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	$90
CBTree (B+tree ISAM driver, multiple variable-length keys)	$80
ME (programmer's editor with C-like macro language by Magma Software)	$75
Wendin PCNX Operating System Shell	$75
Wendin PCVMS Operating System Shell	$75
Wendin Operating System Construction Kit	$75
Entelekon C Function Library (screen, graphics, keyboard, string, printer, etc.)	$70
Entelekon Power Windows (menus, overlays, messages, alarms, file handling, etc.)	$70
EZ_ASM (assembly language macros bridging C and MASM)	$60
Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticomm modem card)	$50
Make (macros, all languages, built-in rules)	$50
Vector-to-Raster Conversion (stroke letters & Tektronix 4010 codes to bitmaps)	$50
Coder's Prolog (inference engine for use with C programs)	$45
PC/MPX (light-weight process manager; includes preemption and cooroutine packages)	$45
Biggerstaff's System Tools (multi-tasking window manager kit)	$40
TELE Kernel (Ken Berry's multi-tasking kernel)	$30
TELE Windows (Ken Berry's window package)	$30
Clisp (Lisp interpreter with extensive internals documentation)	$30
Translate Rules to C (YACC-like function generator for rule-based systems)	$30
6-Pack of Editors (six public domain editors for use, study & hacking)	$30
ICON (string and list processing language, Version 6 and update)	$25
LEX (lexical analyzer generator)	$25
Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor)	$25
C Compiler Torture Test (checks a C compiler against K & R)	$20
PKG (task-to-task protocol package)	$20
A68 (68000 cross-assembler)	$20
Small-C (C subset compiler for 8080 and 8088)	$20
tiny-c (C subsubset interpreter including the tiny-c shell)	$20
Xlisp 1.5a (Lisp interpreter including tiny-Prolog in Lisp)	$20
List-Pac (C functions for lists, stacks, and queues)	$20
XLT Macro Processor (general purpose text translator)	$15
C Tools (exception macros, wc, pp, roff, grep, printf, hash, declare, banner, Pascal-to-C)	$15

Data

Protein Sequences (roughly 4,000 protein sequences with similarity search program)	$60
Webster's Second Dictionary (234,932 words)	$60
U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	$35
The World Digitized (100,000 longitude/latitude of world country boundaries)	$30
KST Fonts (13,200 characters in 139 mixed fonts: specify TEX or bitmap format)	$30
NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	$15
U. S. Map (15,701 points of state boundaries)	$15

The Austin Code Works

11100 Leafwood Lane

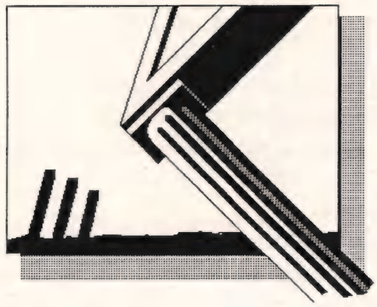
Austin, Texas USA 78750-3409

(512) 258-0785

Free surface shipping on prepaid orders **MasterCard/VISA**

CIRCLE 250 ON READER SERVICE CARD

THE FORTH COLUMN



With this issue, DDJ *both inaugurates a new column and welcomes a new columnist to its ranks. Martin is eminently qualified to discuss the issues and nuances of the Forth language: he is one of the founders of MicroMotion, a senior programmer at FORTH Inc., and vice-president of the Forth Interest Group (FIG).—eds.*

Sure, you say, a column on Forth programming makes sense, but what's this business about news and reviews? Well, a lot has been happening lately. Forth is being used in everything from digital signal processing to neural nets. There are several Forth conferences each year (one near you) and hundreds of publications. You can find Forth in credit-card phones, cyclotrons, and on board the *Titanic*. Staying current can be a major task, but I'll try to keep you well informed and up to date.

There are four sources of Forth information that you should know about: the Forth Interest Group, The Institute for Applied Forth Research, various Forth electronic bulletin boards, and (of course) *DDJ*.

Forth Interest Group

The Forth Interest Group (FIG) brings together more than 4,000 Forth programmers and hobbyists in more

by Martin Tracy

than 80 local chapters. You can order just about any Forth publication through FIG, or find a job, or purchase group health insurance. FIG sponsors both the National Forth Convention and the Forth Modification Laboratory (FORML) conference. Membership is $30/year and includes a subscription to *Forth Dimensions*. You can join by calling the FIG hot line: (408) 277-0668.

The National Forth Convention is where you go to meet your local vendor and to learn more about FIG. More than 40 vendors attended last year, many with display booths. You can hear panels of Forth celebrities speculating about the future of the language. There are two days of technical presentations and a banquet, too. This year's convention meets November 14–15 at the Red Lion Inn in San Jose, California. The National Forth Convention is usually held one or two weeks before FORML so that visitors from abroad can reasonably attend both.

FORML

Imagine spending Thanksgiving this year in beautiful Monterey, California. FORML is *the* technical Forth conference and is held every year following Thanksgiving (November 27–29) at Asilomar in nearby Pacific Grove. Asilomar is a modern conference center situated in a pine forest overlooking the Pacific Ocean. There are raccoon and deer in abundance, and wine and cheese parties are held nightly. Although the technical content of FORML is quite advanced, the atmosphere is relaxed and informal.

The theme of this year's ninth FORML is "Forth and the 32-bit Computer." The call for abstracts (100 words or less) for the ninth annual FORML is September 1, 1987, so you've just missed it. Completed papers are due November 1, which means that if you call right away you might still be able to sneak in on time. You can call the organizers at (408) 277-0668.

The New Forth Dimensions

Actually, its pretty much the same old *Forth Dimensions* but with a new look that includes a glossy cover and a snappier format. *Forth Dimensions* is the bimonthly publication of FIG, and a one year's subscription is included with each annual membership. Some even say that FIG membership is free when you subscribe to *Forth Dimensions*. Volume IX, Number 1, starts the new look with articles on fractal landscapes, 32-bit Forths, and other topics. Marlin Ouverson, the editor, keeps a careful balance of beginning and advanced material, with tiny thermometers printed next to each to show you the level of difficulty.

This same issue includes an extended interview with Elizabeth Rather entitled "Starting FORTH Inc." If you want to know how it all got started, here's where to find out all about it. You might also be interested in Ray Duncan's series "Starting Your Own Software House" in *Programmer's Journal*. Ray is known to *DDJ* readers for his 16-bit Software Toolbox column, but he is also well known to the Forth community as the founder of Laboratory Microsystems Inc. (LMI) and the creator of PC/FORTH.

The Forth Model Library

The Forth Model Library is a series of selected Forth programs published by

What experts are saying about PC Scheme from Texas Instruments:

"...Complete, well-designed and inexpensive at $95 ...a complete Lisp implementation for the professional programmer."

PC Tech Journal, August 1986 (Product of the Month)

Discover how powerful—and inexpensive—PC symbolic programming can be with PC Scheme from Texas Instruments. Whether you're an experienced Lisp programmer or just beginning, PC Scheme is the complete, $95* solution to your software development needs.

PC Scheme combines elegant simplicity with remarkable speed in a full Lisp development system. Named *PC Tech Journal*'s Product of the Month (August 1986), PC Scheme brings professional Lisp programming features to personal computers.

PC Scheme 3.0

—Optimizing incremental byte-code compiler for ease of programming and operation
—EMACS-like editor
—Lexical scoping of variables
—Ability to suspend PC Scheme, execute DOS-based programs, then return to PC Scheme
—Random-file access and binary-file support
—Extensions for debugging, graphics and windowing
—External language interface to C, Turbo Pascal® and other languages
—SCOOPS (Scheme Object-Oriented Programming System)
—Two-megabyte extended/expanded memory support
—New manuals with tutorials and examples

Find out for yourself why experts are praising PC Scheme. For the dealer nearest you, or to order by phone, call toll-free:

1-800-527-3500

* *TI Suggested list price*

PC Scheme runs on IBM® Personal Computers and compatibles (including the Texas Instruments Business-Pro™ computer). Minimum configuration: 512K RAM, dual floppy system.

Turbo Pascal is a registered trademark of Borland International. IBM is a registered trademark of International Business Machines Corporation. Business-Pro is a trademark of Texas Instruments Incorporated.

TEXAS INSTRUMENTS

261933B
© 1987 Texas Instruments Incorporated

(continued from page 132)

FIG and written in the Forth-83 standard dialect. The library is, in part, an ongoing experiment to determine the ability of the Forth-83 standard dialect to support substantial applications. Each participating vendor supplies an appropriate "front end" so that the application can run under its Forth. Needless to say, no machine-code words are used in the library. The emphasis is on readability and utility rather than on speed. Of course, you can always tune it more closely to your Forth or to your needs, hence the name *model*.

This library currently contains five volumes: *A Forth List Handler*, Volume 1, by Martin J. Tracy; *A Forth Spreadsheet*, Volume 2, by Craig A. Lindley; *Automatic Structure Charts*, Volume 3, by Kim R. Harris; *A Simple Inference Engine*, Volume 4, by Martin J. Tracy; and *The Math Box*, Volume 6, by Nathaniel Grossman. Volume 5, *A Complete PROLOG*, by Lou Odette, has been delayed but should appear shortly.

The Forth Model Library is available from FIG at $40 per volume. It is available on IBM MS-DOS 5¼-inch disks and runs under several IBM PC Forths, including Laxen/Perry (public domain) F83; PC/FORTH, Version 3.0 or later; MicroMotion MasterForth, Version 1.0 or later; and FORTH Inc.'s polyFORTH II MS-DOS ISD-4.

Fixed-Point Math

The Math Box, Volume 6 of the library, includes the source code for extended double-precision arithmetic; a complete fixed-point math package; auto-ranging text graphics; and utilities for rapid polynomial evaluation, continued fractions, and Monte Carlo factorization.

Forth programmers generally prefer fixed-point to floating-point math. For any given number size, fixed-point math is faster and more accurate than floating point but it lacks the convenience and larger dynamic range of floating point. Real-time applications that read transducers and write to D/A converters or stepper motor controllers usually have well-understood algorithms of limited dynamic range. PID controllers, digital filters, and computer graphics are especially amenable to fixed-point solutions.

A common Forth approach to implementing fixed-point numbers is to combine signed 16-bit integers and signed 14-bit fractions. A 14-bit fraction is a signed 16-bit number with the binary point two positions from the left:

```
S#.## #### #### ####
```

where *S* is the sign bit and each # is a binary digit. Fourteen-bit fractions have several charming properties:

- They can be added to each other with no adjustment.
- They can be multiplied by an integer with no adjustment.
- They can be multiplied together and adjusted with two left shifts.
- They can exactly represent +1 and −1, which is especially useful when working with sines and cosines.
- Because the number of bits is even and fixed, the square root is exact and

HIGH SPEED FORTH/C PC-RISC SYSTEM

FASTEST FORTH DEVELOPMENT SYSTEM

- Add up to 40 MIPS of power to your IBM PC/XT/AT
- Each PC4000 plug-in card runs at 5 to 8 MIPS
- Fast Forth development and execution in hardware
- More than twice as fast as VAX-11/780 or 68020
- Parallel processing using multiple PC4000s
- NC4016 RISC Engine CPU with 512K on-board RAM
- Ideal for process control or image processing

FASTER C DEVELOPMENT ENVIRONMENT

- 4 times faster than Microsoft C v. 4.0
- K & R Standard C with new extensions
- Compiler, assembler, linker and editor
- 128K bytes of address space for data
- Supports in-line assembly code

- PC4000 : $1,295
- SCForth : $195
- SC-C : $595
- Assembler: $295

SILICON COMPOSERS
210 California Ave.
Palo Alto, CA 94306
(415) 322-8763

SILICON COMPOSERS

CIRCLE 358 ON READER SERVICE CARD

9-Track Tape Subsystem
for the IBM PC/XT/AT
XENIX or
MS-DOS.

The solution to your micro/mainframe communications problem is available today!

Qualstar's new ½ inch 9-track MINISTREAMER™ brings full ANSI data interchange capability to the PC. Now you can exchange data files with virtually any other computer using 9-track tape.

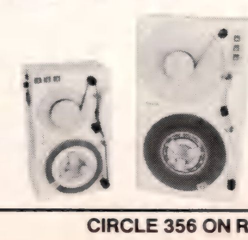
Available in both 7" and 10½" versions, the MINISTREAMER weighs in at only 27 pounds and uses less desk space than an ordinary sheet of paper, yet provides full 1600/3200 BPI capability at an affordable price. Up to 134 megabytes of data (depending on format) can be stored on a standard 10½" reel of tape, thus making the MINISTREAMER a highly-reliable answer to your backup requirements as well.

Tape subsystem includes tape drive, coupler card, cables, dust-cover and MS-DOS or XENIX compatible software. Prices start at $2,995.

386 READY!

Discover the many advantages 9-track tape has over other Micro/Mainframe links.

Call us today!



QUALSTAR®

9621 Irondale Avenue,
Chatsworth, CA 91311
Telephone: (818) 882-5822

CIRCLE 356 ON READER SERVICE CARD

does not require an extra multiplication.

You can read more about 14-bit fractions in Leo Brodie's *Starting Forth*, 2d ed (Englewood Cliffs, N.J.: Prentice-Hall, 1987). Don't bother trying to find the information in the first edition—it isn't there. By the way, *Starting Forth* has sold more than 110,000 copies. The second edition has been substantially revised and enlarged. It now identifies the differences between Forth dialects and includes a long-awaited index.

Unfortunately, neither 16-bit integers nor 14-bit fractions can represent handy numbers such as 3.1415 . . . . To combine efficiency with convenience, Dr. Grossman has designed a fixed-point package based on signed, 32-bit, fixed-point numbers with the binary point in the middle:

```
S### #### #### #### #### #### ####
                                ####
```

These numbers can range from more than 9,999 to less than 0.0001 with roughly 4½ digits on each side of the decimal point.

The 1987 Rochester Forth Convention

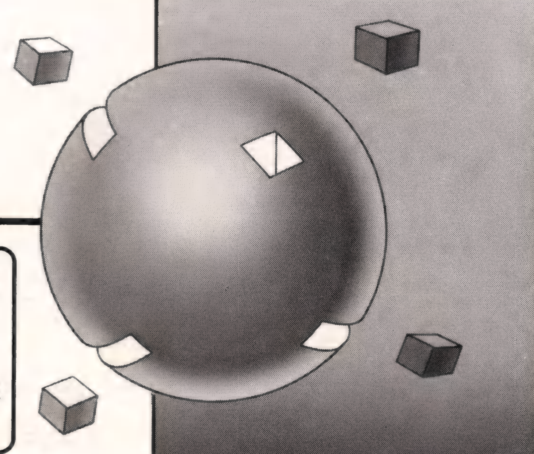
This year's Rochester Forth Convention was an impressive gathering of more than 100 Forth professionals from industry and academia. Although it was difficult for me to sit through 50 technical presentations (I admit I missed some), the four-day conference sped along, accompanied by excellent food and nightly parties. This year's conference, sponsored by the Institute of Applied Forth Research and coordinated by Larry Forsley, was a bit less organized than last year's, sadly because of Thea Martin's disinvolvement. Thea is leaving the institute to pursue a career in education.

This year's official theme was Forth hardware architecture, and the unofficial theme was artificial intelligence. Dr. Dorband (NASA Goddard Space Flight Center, Maryland) talked about the Massively Parallel Processor (MPP). This 128 × 128 array of 16,384 serial processors is programmed in (can you guess?) Forth. Dr. Ting (Lockheed) described an NCR GAPP network with a Novix 4016 as the microcode sequencer.

On a slightly smaller scale, Phil Koopman, Jr., demonstrated his Writable Instruction Set/Stack Oriented Computer (WISC), a 32-bit commercial Forth engine. Wright State University (Ohio) presented a stack-frame computer designed for Forth. Final silicon is expected in spring 1988. The university's stack-frame architecture has a shallow stack with all items directly addressable. ROT would be faster on this stack-frame computer than on, let's say, a Novix 4016. Speaking of Novix chips, George Nicol had an IBM AT stuffed with six Silicon Composer PC4000 boards running in parallel. Let's see, by my count that's 20 MIPS per board, for 120 MIPS. Is that Million Instructions Per Second or Misleading Information Provided by Sales? Also speaking of Novix chips, watch for good things from Harris Semiconductors, which has added the Novix to its macro cell library.

On the AI front, the University of Utah has been using an emulator written in Forth to test its Common

C PROGRAMMERS!
THE TOOLS YOU NEED AT A PRICE YOU'LL LIKE



BTree Supports all index file operations. Very quick sequential or random access, duplicate keys, multiple indices, fixed and variable length data records are all supported. **75.00**

ISAM Works on top of BTree to provide a simple, yet powerful application program/file system interface. Complex filesystem manipulation becomes a snap. Provides the power of a database manager with the flexibility of a programming language. **40.00**

NOW MULTI-USER AVAILABLE 60.00

Ip Finally, a completely device independent printer library! Ip drives any printer as accurately as possible and allows easy access to its most sophisticated features. Multiple fonts, multi-column output, complex margin formatting, and much more. Pays for itself the first time it's used. **75.00**

Snake The ultimate 'make' utility. We couldn't find a good one, so we wrote a great one. Has all kinds of powerful features including wild card filename expansion, nested macros, and multiple dependency and rules definitions. Ready to go for MS-DOS; C source is there if you use another operating system. **59.00**

Combine & Save: BTree + ISAM + Ip **159.00** + Snake **199.00**

Each product includes a typeset manual, example programs, and complete C source code that runs on any operating system. Softfocus products may be incorporated into applications royalty-free.

Credit card orders accepted. Visa, M/C, Amex. Dealer inquiries invited.

softfocus
1343 Stanbury Drive
Oakville, Ontario, Canada
L6L 2J5
(416) 825-0903

CIRCLE 259 ON READER SERVICE CARD

It's good for your system! **Vitamin C**

"If you need source code, make sure your wallet is wide open or get VITAMIN C.

Picking the best value package is hard... If you're a source code fanatic like me, VITAMIN C is preferable."

- Computer Language, June, 1987

Fast, flexible, versatile, reliable. Vitamin C delivers the vital combination software professionals demand to produce superior applications in dramatically less time. Highly efficient, professionally crafted C code provides lightning fast displays required by today's window intensive programs.

High level functions provide maximum productivity and require little supporting code. Extended versions of these routines add flexible control over specific details when necessary. Plus, Vitamin C's versatile, open ended design is full of hooks so you can intercept and plug-in special handlers to customize or add features to most routines.

VCScreen, our screen painter / code generator speeds your development even more! Simply draw your input forms using our interactive design editor and generate perfect C source code ready to compile and link with the Vitamin C library.

- Professional C function library
- 30 day money back guarantee
- Multiple bullet proof windows
- Easy full screen data entry
- Unlimited data validation
- Context sensitive help manager
- Menus like Lotus and Mac
- Programmable keyboard handler
- Text editor routines
- No royalties or runtime fees
- Library source included FREE
- Free technical support
- Free BBS at (214)418-0059
- Supports all major compilers including Microsoft 5.0
- VCScreen code generator too!
- UNIX version avaialable, call for details

Vitamin C $225
Includes all source code FREE! For IBM PC, XT, AT, PS/2 and true compatibles. Specify compiler when ordering.

VCScreen $99.95
For IBM PC, XT, AT, PS/2 and true compatibles. Requires Vitamin C.

We ship UPS. Please include $3 for ground, $6 for 2-day air, $20 for overnight, or $30 if outside the U.S. Texas residents add 7 1/4 % sales tax. All funds MUST be in U.S. dollars drawn on a U.S. bank. Visa & MasterCard accepted.

ORDER NOW!
(214)416-6447

creative PROGRAMMING
Box 112097 • Carrollton, Texas 75011

Windows • Data Entry • Menus • Help • Text Editing
Plus... All Source Code FREE!

THE FORTH COLUMN
(continued from page 135)

LISP compiler. Ecole Polytechnique de Montreal presented FUZZY-FORTH, a real-time fuzzy-rule production system. Henry Harris (Jet Propulsion Labs, California) gave a too-short talk on conceptual dependency. William Dress (Oakridge National Labs, Tennessee) brought a bug! Complete with a few hundred simulated neurons connecting simulated sensors with simulated muscles, this artificial neural network "insect" crawled around a CRT, learning to avoid the sides and to seek the simulated food. There were talks on computer-aided medical diagnosis, multiprocessor expert systems, and several other aspects of AI.

The last day of the conference was reserved for seminars and demonstrations. The peripatetic Ray Duncan demonstrated LMI's UR/FORTH for the Microsoft OS/2 operating system. Again and again (remember Creative Solution's MacFORTH?) Forth is one of the first languages to run in a new software environment.

If you missed the conference, you can still read the proceedings. They will appear shortly as a special issue of the *Journal of Forth Applications and Research* (*JFAR*). A one-year subscription (four issues) to this professional, refereed publication costs $40. You can order it from *JFAR*, P.O. Box 27686, Rochester, New York 14627.

For a Good Time . . .

If you own a modem, there are two numbers you should definitely investigate: the West Coast Forth Board ([213] 301-0761) in Los Angeles and the East Coast Forth Board ([703] 442-8695) in McLean, Virginia. These public-spirited nondenominational Forth bulletin boards contain megabytes of Forth utilities, interviews, contacts, and news. The WCFB sysops are Scott Squires (Lucasfilm), Michael Ham (of *DDJ* fame), and the ubiquitous Ray Duncan. The ECFB sysop is Jerry Shifrin (MCI).

Both boards are PCBOARD Forth-only bulletin-board systems. The ECFB is the more active of the two, averaging about two dozen messages a day. There are more than 1,000 Forth files available on this board! You can download the complete di-

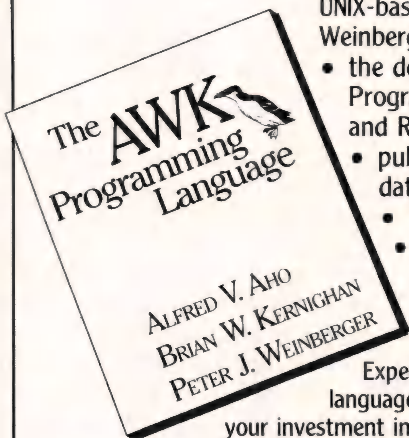
AWK

The Software

MKS AWK

- the **only** complete version of AWK available for DOS;
- **fully compatible** with the latest description in **The AWK Programming Language**, by Aho, Weinberger, and Kernighan;
- easy to learn, giving beginners results with little effort;
- the natural introduction to mastering the C programming language;
- text substitution and pattern matching;
- definable functions;
- associative arrays and regular expressions;
- sample programs and a tutorial manual;
- large model version;
- hardware floating point version;
- rapid prototyping tool for larger programs.

The Book



- written by the authors of the original UNIX-based program, Alfred V. Aho, Peter J. Weinberger, and Brian W. Kernighan;
- the definitive book on AWK just as "The C Programming Language," by Kernighan and Ritchie, defined C;
- published by Addison-Wesley (release date: September 1987);
- lists at $21.95;
- with MKS AWK only $14.00.

Experience the power of a UNIX programming language on your desktop PC without sacrificing your investment in DOS applications and training.

AWK is a versatile first language for non-programmers and a sophisticated data retrieval and report generation tool for the experienced user. Based on a sequence of terse pattern/action rules, AWK allows you to manipulate files for retrieval, transformation, reduction, and validation of data. MKS AWK comes with full technical and tutorial documentation to speed your mastering of this fourth generation programming language.

MKS AWK sells for **$75.**

Order both the software and the book from MKS for **$89.**

Mortice Kern Systems Inc.,

43 Bridgeport Road East, Waterloo, Ontario, Canada, N2J 2J4 **(519) 884-2251**

uucp: {allegra, decvax, ihnp4}!watmath!mks!toolkit

MKS AWK runs under MS-DOS 2.0 or later. Not copy protected. Prices quoted in US funds. VISA, MASTERCARD, American Express, uucp, and purchase orders are accepted. Overseas orders please add $10 for postage and handling. MKS is a registered trademark of Mortice Kern Systems Inc. UNIX is a trademark of AT&T Bell Labs. MS-DOS is a trademark of Microsoft Corp.

CIRCLE 249 ON READER SERVICE CARD

(continued from page 137)

rectory listing in FILELIST.ARC. Many of these files are also available on the WCFB. Jerry Shifrin has put together an excellent 50-page *Guide to the ECFB*, which is available by leaving him a message on the board.

Product of the Month: Ashton-Tate's RapidFile

The following Forth success story was downloaded from the ECFB. Editorial comments are by Jerry Shifrin:

"Ashton-Tate has recently released RapidFile, a data management package for the IBM PC. The following is excerpted from Ashton-Tate's 3/87 issue of *Technotes*:

Technotes: Why is RapidFile written in FORTH?

Mark McDonough (developer): I'll tell you the story. When I told my professors that I wanted to implement the QBE [IBM's Query By Example] concept on a PC, I was told it was not possible. Knowing that it couldn't be done, I went out to find a genius who would know how to push the PC to its limits. I found that genius in Tom Dowling [of Miller Microsystems].

Tom claimed, and has since proven time and again, that FORTH was as easy to program in as other high level languages and that its programs run almost as fast as assembler programs. FORTH programs also compile into a very small space. We could not have fit RapidFile onto one disk and in 256K of RAM if it were written in C.

Technotes: We understand that the flexibility of FORTH really came in handy when designing RapidFile.

Steve Allin (product manager): Yes, FORTH is very tailorable. It allowed us to make substantial changes to the design. It is very close to being a prototyping language where you can make major changes much more easily than in standard programming languages, which would have required substantial rewrites. RapidFile went through several versions before its release and it couldn't have gone through all that if it hadn't been written in FORTH."

The ANS Forth Effort Begins

In October 1986, a small group from the Forth community met at FORTH Inc., Los Angeles, to submit a proposal to the American National Standards Institute (ANSI) asking it to undertake a Forth standard. This group consisted of major vendors (Elizabeth Rather, FORTH Inc.; and Ray Duncan, Lab Microsystems Inc.); major users (W. B. Dress, Oakridge National Laboratory; Burt Felis, IBM Corp.; and Jerry Shifrin, MCI Telecommunications Corp.), and interested experts (Charles Moore, inventor of Forth; Greg Bailey, Athena Programming; and yours truly). Also invited but unable to attend were Don Colburn (Creative Solutions), Dick Miller (Miller Microcomputer Services), and Kim Harris (Paradise Systems).

In February 1987, CBEMA (the ANSI organization responsible for ANS FORTRAN and the proposed ANS C), sent the ANS Forth proposal X3J14 to its general membership for a letter ballot. In April, CBEMA accepted the ANS Forth proposal. The following is from a letter by Ms. Rather to members of the proposing group:

"I am happy to report that the final vote for the establishment of X3J14, the Technical Committee for ANS Forth, was favorable: 36—1—1, with 2/3 approval required. An organizational meeting has been scheduled for August 3—4, 1987, at CBEMA headquarters, 311 First St., N.W., Washington, D.C. A newsletter to all ANSI members and a press release will be sent out May 1.

"At the first meeting, the staff of the X3 Secretariat will present a detailed tutorial on X3 and TC policies and procedures. Our first assignment will be to develop a detailed work plan and schedule for development of a draft standard."

Anyone may attend this meeting. According to ANSI rules, however, a voting member of a technical committee pays a yearly fee of $175 and must attend at least two out of three meetings to retain voting status. Furthermore, a technical committee (TC) generally meets two to four times a year in fairly distributed geographic locations. The frequency of meetings is part of the work plan decided at the first meeting. You can apparently team up with an alternate and thereby attend every other meeting without losing your vote.

While the direction of ANS Forth is yet to be determined by the yet-to-be-determined TC, the formal proposal calls for an integration rather than a revolution. The first item in the proposed work program is to "identify and evaluate common existing practices in the area of the Forth programming language." Wil Baden, former ANS FORTRAN committee member and well-known Forth orator, calls this "the principle of least surprise." Furthermore, the proposal continues, "while the Forth-83 Standard has stabilized the language to a great extent, it has proven too restrictive and machine-dependent. Assuming the ANS Forth standard confines itself to such changes as are necessary to resolve the problems in Forth-83, the effect on current practice will be modest." It was generally agreed that an ANS Forth might be a nice place to delete all references to a 16-bit parameter stack.

Meanwhile, Guy Kelly, chair of the Forth Standards Team (FST), has suggested that FST might serve as a clearinghouse for proposed extensions to Forth, such as string operators and floating-point arithmetic. Technical proposals should be sent to the Forth Standards Team, P.O. Box 4545, Mountain View, CA 94040. There is a form provided at the back of every published Forth-83 Standard. FST developed the Forth-79 and Forth-83 standards but has no further meetings scheduled at this time.

At any rate, by the time you read this, the first meeting will already have happened. I will try to keep you up to date on this heroic and historic event.

Coming Next

Next time around? More on the ANS Forth effort, a new bibliography on Forth, and a few surprises.

DDJ

Vote for your favorite feature/article.
Circle Reader Service **No. 7.**

Ask for a FREE REPORT: "Interviews with Authors & Users:
How Can These Products Raise Productivity and Help You Write Better Programs?" from

THE PROGRAMMER'S SHOP

ACCELERATE YOUR C DEVELOPMENT
*TRANSLATE** FORTRAN TO C

- **Maximize the vast resources of FORTRAN** while moving up to C. Speed up new C development and avoid reinventing the wheel.
- Use ***RTC Plus*** to translate FORTRAN code and libraries — and **maintain code with greater ease and flexibility** in C.
- **Source code to C libraries is included.**
- ***RTC Plus* supports standard FORTRAN-77** as well as some DEC VAX extensions (excluding FORTRAN I/O, character and complex statements/expressions). Over 95% of STUG's RATFOR is also supported. ***RTC Plus*** generates K&R C.
- **Finally a cost-effective method of conversion into C.**

ORDER TODAY!

RTC Plus

DEMO $10
MS-DOS $450
PS PRICE $399

****Translate: "To convey to heaven without natural death."***

COBALT BLUE

1683 MILROY, SUITE 101, SAN JOSE, CA 95124
408-723-0474

CIRCLE 301 ON READER SERVICE CARD

Getting started in expert system development has never been this *Easy*

Announcing the Personal Consultant™ Series, from Texas Instruments. Now there's a family of powerful expert system development tools to get you started and keep you going.

Personal Consultant *Easy* (PC *Easy*) runs on select members of the TI Professional and IBM® Personal Computer families or compatibles. Designed for those just getting started in expert system development, PC *Easy* is the low-cost, high-functionality tool for rapid prototyping and development of expert system applications on personal computers for only $495.*

Personal Consultant *Plus* (PC *Plus*), the larger, more powerful member of the Personal Consultant Series, is priced at $2,950.* Designed to take advantage of today's more powerful AT-class of personal computers, PC *Plus* provides extended knowledge representation features; increased rule capacity; and access to the Lisp language allowing sophisticated developers the flexibility to extensively customize their applications.

Both microcomputer products feature a powerful rule entry language with integrated window-oriented editor; comprehensive user explanation facilities such as WHY, HOW, HELP, and REVIEW; support for TI and IBM EGA graphics; access to external information through DOS files or dBase™ inquiries; and the ability to deliver cost-effective versions of your applications with the addition of an optional run-time diskette.

Knowledge bases created using PC *Easy* are 100% upwardly compatible with the higher functionality PC *Plus* product on a microcomputer, allowing you to "get started and keep going" with total confidence that your software investment will be preserved.

PC Easy	**$ 439**
PC Plus	**$2599**

TEXAS INSTRUMENTS

*TI List Price, subject to change without notice.
Personal Consultant and Explorer are trademarks of Texas Instruments Incorporated.
IBM is a registered trademark of International Business Machines Corporation.
dBase is a trademark of Ashton-Tate.

CIRCLE 302 ON READER SERVICE CARD

©1986 TI 261765-04A

Now you see it; Now you don't!
TSR's made easy with /*resident_C*/

Finally! You write your C program and we make it resident. No strings attached. Want to process interrupts also? No problem. **/*resident_C*/** is a set of C functions that enable you to process interrupts and/or make your programs "terminate stay resident" like Sidekick.

We've done the research, testing, and grunt work to make your TSR programs safe, compatible and easy. Be aware of the "ifs, ands and buts" when writing interrupt handling software. Only **/*resident_C*/** has all that you need to write reliable interrupt handlers without the worries.

We don't ask you to trust us either. Our manual explains what we are doing very clearly and our documented source code is available. In addition, our working demos give you clear examples of resident programs, interrupt handlers and resident libraries. There is no other product that can do what **/*resident_C*/** can do for you.

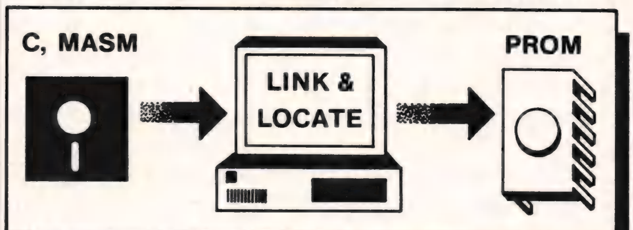
/* resident_C */ is the perfect complement to the other library products available from Essential Software.

ESI

List: $99 / with Source add $99
PS: $79 / with Source add $79

CIRCLE 303 ON READER SERVICE CARD

FIRMWARE DEVELOPMENT



LINK & LOCATE enables PC users to produce ROM-based firmware for 8086/87/186 from object files generated by popular C compilers, such as from Microsoft, Lattice and Borland's Turbo C, and MASM from Microsoft, Provides full control of segment placement anywhere in memory. Supports output of Intel HEX file for PROM programmers, Intel OMF absolute object file for symbolic debuggers and in-circuit emulators. Includes Intel compatible linker, locator, librarian, hex formatter and cross reference generator. $350.

NEW! Includes utility to support PC based PROM programmers.

SYSTEMS & SOFTWARE INC.

PS Price: $315

3303 Harbor Blvd., C11, Costa Mesa, CA 92626
Phone (714) 241-8650 FAX (714) 241-0377 TWX 910-695-0125

CIRCLE 304 ON READER SERVICE CARD

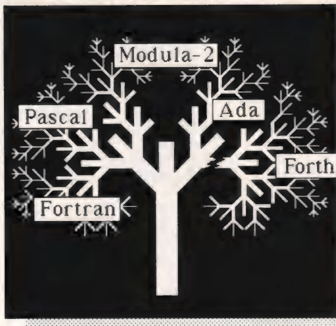
Call or Circle Reader Service Number for Your FREE Study.

HOURS: 8:30 A.M.-8:00 P.M. E.S.T.
5-D Pond Park Road, Hingham, MA 02043
Mass: 800-442-8070 or 617-740-2510 /87

800-421-8006

THE PROGRAMMER'S SHOP
Your complete source for software, services and answers

Data Hiding and Its Variations



To trench-level programmers, the concept of data hiding is likely to be regarded with hostility—why deny full access to data structures? Unfortunately, data hiding is part of the baggage of modern software engineering, playing an important role in minimizing delays in application development. Because teams of programmers may spend thousands of man-hours writing code, strict control procedures can be vital to the successful management of complicated programming projects.

For example, when using either Ada or Modula-2, the first step in software coding begins with writing the definition modules, or packages. These modules list the exported data types, constants, variables, and routines. They will form the foundation upon which each team will develop the actual detailed modules and packages.

Careful planning must be exercised in the planning phase both to define and confine software development objectives. Each team should be able to work as independently as possible, thus yielding maximum productivity with minimum confusion. In the absence of data hiding, all exported data types and their variables have visible structures; this gives any team the ability to write their own routines and to manipulate the exported data structures.

by Namir Clement Shammas

But what happens if the team responsible for implementing a specific module and its data structures discovers a more efficient equivalent structure or decides that it absolutely has to modify the current one? The result is a domino effect of recoding by all the teams that used those modified data types. Keep in mind that this, too, is likely to create an entirely new set of problems!

To prevent this disastrous domino effect, opaque data types can be used. This gives the module or package implementor the freedom to alter the data structure without passing the negative effects on to other programmers. So what's the price for this?

Opaque data types must be accompanied by an adequate set of exported basic routines to manipulate them because client modules and programs cannot contain their own routines to access the components of an opaque type. This means that informed choices must be made in specifying which routines accompany the opaque types.

Data hiding is used with data structures that have alternate representations, such as:

- complex numbers represented by rectangular or polar components
- stacks implemented using arrays, a set of scalar variables (for small stacks), or a linked list
- lists implemented using pointers to dynamic data or to arrays
- binary trees implemented using pointers to dynamic data or to arrays
- two-dimensional matrices represented by arrays of rows or arrays of columns

Although data hiding is formally implemented in languages such as Ada and Modula-2, it can also prove useful to emulate this feature fully, or even partially, in other languages. This article looks at example applications in BASIC and Pascal and compares these implementations of data hiding to those in Ada and Modula-2.

The extent to which you can emulate data hiding in a particular language/implementation is dependent on two main ingredients: making the detailed data structure opaque, and denying the programmer access to the routines that manipulate the opaque data types. The first requirement is easily met by disguising the data structure; thus the trick is to hide the supporting code.

BASIC Examples

The QuickBASIC implementation produces a compiled user library that meets the more critical code-hiding requirement. You can use either arrays or large strings (QuickBASIC supports strings of up to 32K). Arrays are probably more suitable for tackling purely numeric data structures, whereas strings are more suitable for managing structures with numeric and alphanumeric data.

QuickBASIC supports packing of the basic numeric data types into fixed-length strings. Strickly speaking, QuickBASIC still enables you to access the components of the not-so-opaque data structure, but the schemes used would make such access a wasted effort. I call this type "logically opaque."

Listing One, page 110, shows the QuickBASIC library that implements

HOT GRAPHICS PACKAGE FOR C PROGRAMS.* $39.95

Everything you need to write dramatic graphics effects into your Eco-C88 C programs. Some of the features include:

- Support for EGA, CGA, and Z100
- Over 100 graphics and support functions, many of which are PLOT-10 compatible.
- Many low level support routines reside outside your small model code-data area
- Can write dots thru the BIOS (for compatibility) or to memory (for speed)
- Graphics function help from CED editor available
- World, pixel or turtle color graphics modes
- 47 standard fill patterns, 17 line dashing patterns, Hershey fonts, plus user defineable fill, dash and fonts
- Supports view areas, rotateable fonts, clipping, arbitrary fill areas, extensive error checking, examples, and user's manual.

A must for the graphics enthusiast and a bargain at only

$39.95

***Requires Eco-C88 C Compiler.**

NEW POP-UP WINDOWS FOR YOUR C PROGRAMS.

This windowing library allows you to add pop up windows in your C programs quickly and easily. Use them for help windows, selection menus, error messages, special effects—anywhere you need an attention getter. Just some of the features include:

- CGA, EGA, and monochrome support
- Slow mode option for "flicker" displays
- Control any program that goes through the BIOS
- Use up to 255 windows
- No special window commands; use print f ()
- Resize and move windows
- Custom window titles and borders
- Can be used with ANSI device driver
- Most of window's code-data lies outside small model limits
- Use any of the IBM text or block characters
- User's manual and examples

The Windowing Library requires an IBM PC compatible BIOS and the Eco-C88 C compiler.

ONLY $29.95

HANDY LIBRARIAN MAKES LIFE EASIER.

Now you can combine your modules, functions, and subroutines into your own library for easy link commands. Fully compatible with ANY standard OBJ format files (not just Ecosoft's products). With the Ecosoft librarian, you can:

- Add, delete, and extract from a library
- Get table of contents or index of a library
- Combine libraries, control library page size, use switches for combinations, process complex library requests, use wildcards, and do library directives from command files.
- Complete with user's manual

A valuable addition for any programmer.

ONLY $29.95

THE FIRST PROFESSIONAL 'C' COMPILER FOR UNDER $60.

A C compiler with many ANSI enhancements at an unbelievably low price. The Eco-C88 C compiler has:

- Prototyping (the new type-checking enhancement)
- Enum and void data types
- Structure passing and assignment
- All operators and data types
- A standard library with more than 200 functions (many of which are System V compatible for greater code portability)
- CC and mini-make that all but automates the compile process
- 8087 support (we sense the 8087 at runtime — no dual libraries)
- ASM or OBJ output for use with MSDOS linker
- Tiered error messages — enable-disable lint-like error checking
- Fast compiles and executing code
- Expanded user's manual
- CED full-screen program editor

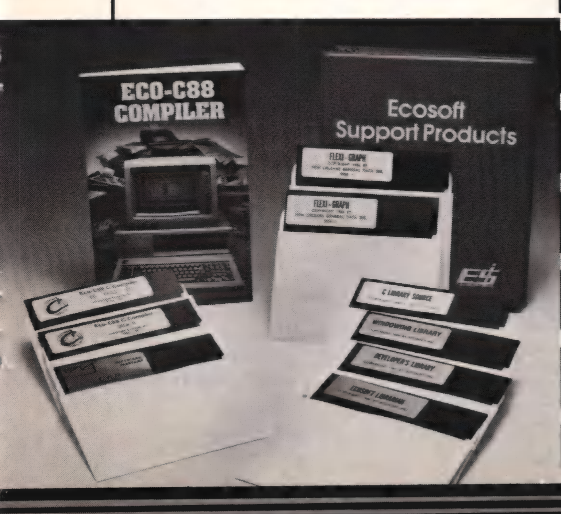
Everything you need at the unbelievable price of $59.95.

Eco-C88 C compiler requires an IBM PC, XT, or AT (or compatible) with 256K of memory, 2 disk drives and MSDOS 2.1 or later.

Orders only:
1-800-952-0472
Technical Information:
(317) 255-6476

NOT COPY PROTECTED.

Ecosoft Inc.
6413 N. College Ave.
Indianapolis, IN 46220



ORDER FORM CLIP & MAIL TO: Ecosoft Inc., 6413 N. College Ave., Indianapolis, IN 46220

ITEM	PRICE	QTY	TOTAL
Flexi-Graph Graphics	$39.95		
Window Library	$29.95		
Eco-Lib Librarian	$29.95		
Eco-C88 C Compiler CED	$59.95		
	SHIPPING		
	TOTAL (IND. RES. ADD 5% TAX)		

PAYMENT: ☐ **VISA** ☐ **MC** ☐ **AE** ☐ **CHECK**

CARD # ______________________ EXPIR DATE __________

NAME ______________________

ADDRESS ______________________

CITY ______________________ STATE __________

ZIP __________ PHONE ______________________

CIRCLE 89 ON READER SERVICE CARD

ECOSOFT

(continued from page 140)

a version of the logically opaque numeric matrix. The matrix is made up of arrays of columns. Three procedures are used: one to initialize the matrix, one to store an element, and one to recall an element.

The scheme works with *OPTION BASE 0* to enable *Mat#(0)* to store the maximum number of rows and columns in the matrix structure. The opaque matrix should be declared as a one-dimensional array with *Max.Row% *Max.Col%* as its upper array bound. The routines are written such that the row and column indices of the opaque matrix start at 1. The routines for storing and recalling matrix elements contain commented code lines for implementing the arrays of rows. In this case, the changes are very simple.

The QuickBASIC code shows how you are still able to access the basic components of the data structure (that is, the individual array elements). With the compiled library version, however, the exact mapping of the matrix elements is invisible.

A second method that can be implemented in both QuickBASIC and Turbo BASIC revolves around static local strings or arrays. Both implementations support static declarations that can be used with local variables, which permits the BASIC subroutines to retain the data between subroutine calls. Example 1, below, shows the general scheme and how the subroutine in question is used to perform multiple tasks. The latter permits the BASIC subroutine to monopolize the opaque data structure. Other BASIC code segments can use the opaque data but are denied direct access. The same methods can be used with C and PL/I because they support static variables.

It's possible to emulate opaque types in Pascal by using pointers.

True BASIC is another BASIC implementation that supports a limited level of data hiding. True BASIC modules support an extension to ANSI BASIC that implements *SHARE* variables, which are characterized by their static nature and by the fact that they are accessible by all the module routines. These features enable programmers to perform partial data hiding.

Listing Two, page 111, shows an implementation of a binary tree in True BASIC. The module implements one instance of a binary tree. Three *SHAREd* arrays are used to manage the binary tree: *Bin_Tree$( )* stores the tree node data, *Left( )* is the array of left pointers, and *Right( )* is the array of right pointers. None of the three arrays are accessible by the application program using the binary tree, which makes the binary tree structure completely opaque.

```
SUB Jekyll.and.Hyde(<argument
      list>, Menu.Choice) STATIC

STATIC <list of scalar and
  arrays used to implement opaque
                       structure>

SELECT CASE Menu.Choice

  CASE 1

      <sequence of statements>

  CASE 2

      <sequence of statements>

  CASE 3

      <sequence of statements>

  ELSE

      <sequence of statements>

END SELECT

END SUB
```

***Example 1:** General scheme for using static local variables in QuickBASIC and Turbo BASIC*

The C Programmer's Assistant

C TOOLSET™

Unix-like Utilities for Managing C Source Code

No C programmer should be without an assistant. C ToolSet provides you with the support you need to make C programming easier.

All of the utilities are tailored to the C language but you can modify them to work with other languages as well.

Source code in standard K&R C is included. You are welcome to use it with any compiler (UNIX compatible) and operating system you choose.

The documentation contains descriptions of each program, a listing of program options, and a sample run. On-line help responds to -? on the command line with a list of options

12 Time Savers

DIFF - Compare files line by line; use CMP to compare byte by byte.
GREP - Regular expression search.
FCHART - Trace the flow of control between large program modules.
PP - Format C program files so they are easier to read.
CUTIL - General purpose file filter.
CCREF - Cross reference variables.
CBC (curly brace checker) - Check for pairing of curly braces, parens, quotes, and comments.
Other utilities include *DOCMAKE, ASCII, NOCOM, AND PRINT.*
Requires MSDOS and 12K RAM

MONEYBACK GUARANTEE

Try C ToolSet ($95) for 30 days — if not satisfied get a full refund.

Call (800) 255-4659
In MA (617) 331-0800

Source™ 541-D Main St., Suite 412, So. Weymouth, MA 02190

CIRCLE 152 ON READER SERVICE CARD

Pascal Examples

Although Pascal is similar to Modula-2, it does not support opaque types. Nevertheless, it is possible to emulate opaque types in Pascal by using pointers. Schneider[1] suggests the following method:

1. Declare an *opaque* record type and its pointer. This record declaration should be empty.
2. Declare a record type with the actual data structure used. Also declare a pointer type accompanying this record type.
3. Declare a variant record of the form:

```
TYPE Convert = RECORD
     CASE boolean OF
        TRUE : (Opaque : <pointer to
                         opaque record>);
        FALSE : (Actual : <pointer to
                         actual structure>)
END;
```

This variant record enables you to pass the addresses of pointers from one record pointer type to the other.

4. Provide two functions for two-way conversion between the opaque type and the actual data structure.
5. Write a set of routines that provide the required manipulation of the opaque data type.

Listing Three, page 113, shows Turbo Pascal data types and routines to implement opaque complex types. Five routines are provided for demonstration. The first two create complex numbers from rectangular or polar coordinates, and the following two perform the conversion in reverse. Function *Add_Complex* is a sample routine to perform a mathematical operation on a pair of complex numbers.

Notice how the input parameters that represent opaque complex numbers are first converted into the actual structures. True addition is performed using rectangular coordinates, and the results are converted into opaque complex numbers. The Pascal code prohibits the programmer from accessing the actual record type. This is enforced even further when using compiled library *UNIT*s because the actual record structure is confined to the implementation part of the library *UNIT*.

Compare the Pascal code with the Modula-2 version in Listing Four, page 116. Notice how simple and elegant the Modula-2 version looks compared to the Pascal version. In both Pascal and Modula-2, you can use the polar coordinates as the actual data structures without changing the parameters of the routines involved.

Availability

All the source code for articles in this issue is available on a single disk. To order send $14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call (415) 366-3600, ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

Note

1. M. Schneider, "Pascal Report," *Journal of Pascal, Ada and Modula-2*, vol. 5, no. 4 (July/August 1986).

DDJ

(Listings begin on page 110.)

Vote for your favorite feature/article.
Circle Reader Service **No. 8**.

THE 150% SOLUTION FOR SUPERIOR DATABASE DEVELOPMENT AT 62% OFF.

PHACT-manager™ gives MS-DOS™ programmers the ISAM they need. Plus a Report Writer, Query Language and full C source code.

- Efficient B + Tree access method.
- Unlimited number of keys and variable length records.
- Security: Password protection, shared/exclusive use.
- Runs on networks.
- Sequel-like query language for interactive or batch query/update.
- Report Writer: Perform "joins," create and use variables, sort, format and more.
- Versions for all popular C compilers.
- Thousands of licenses sold.

To order or get more information, call us at
1-800-222-0550 (outside NJ) or 1-201-985-8000 now.
MasterCard/Visa accepted.

Only $249 complete! Full C source code. No royalties!

UniPress Software

UniPress Software, Inc. 2025 Lincoln Highway Edison, NJ 08817

MS-DOS is a trademark of Microsoft. PHACT-manager is a trademark of PHACT Associates.

CIRCLE 77 ON READER SERVICE CARD

(continued from page 14)

posed of integers, unless you're a throwback to the Pythagoreans. Because of the nature of the problems people need to solve today, modern computer languages use sophisticated variables or object definitions, noninteger arithmetic operations, and advanced programming control techniques. Because computers do not support these functions, they must be emulated somehow by languages or programmers or both, and emulation is just a fancy word to describe a case in which a particular machine is made to do something it was not meant to do.

The exception to this is assembly languages, which are fully supported by the computers they are written for, but as anyone knows who's done enough "real programming," assembly language isn't the easiest language to program in, document, or maintain.

Therein lies the problem with computers and the reason why there are no good computer languages today and certainly no good general-purpose computer languages. Trying to tailor a machine that does one particular job well to do another job always leads to inefficiency, but instead of changing the approach, most people try and improve on just one aspect of the problem—the software.

We will not see any significant advance in the way we program computers until a computer architecture appears that supports a particular high-level language so fully that it would take more time and be less readable and maintainable to program in assembly language than to use the HLL. This architecture should be geared toward a particular type of task, such as object-oriented programming, real-time systems, or arithmetic number crunching, and should not support any operations that are not needed. Thus, we'll see two or three different types of computers, all with architectures supported by the right language to make the machines easy to program and as swift as possible.

People invent new languages to make the job easier, looking at the problem strictly from a software aspect. Instead, they should take a systems approach and think about the whole system—hardware and software—and produce a machine that gives the optimal solution to a particular type of problem. The new generation of computers should clear away the vast jungle of computer languages by introducing machines that are so tailored to one particular language that to use another one would be grossly inefficient. These new machines would not necessarily run any of the languages that are currently available. In fact, they probably won't. This doesn't mean that all computers will look like clones of three or four basic types. There will still be a lot of latitude in what type of I/O devices a particular system will support for a particular job or individual. But the computers will be efficient machines especially tailored to the task to be done, instead of Fords or Chevys passing off as Mercedes or BMWs.

There are microprocessors geared to run a particular language, but not everybody is convinced that programming in Forth or Modula-2 is the way to go, and in any case the problem should be approached from a system standpoint instead of just optimizing the microprocessor.

David Nakamoto
280 S. Euclid Ave., #315
Pasadena, CA 91101

Polytron Responds

Dear *DDJ*,

In the C Chest column in the May 1987 issue, Alan Holub briefly discussed the relative merits of Polytron's PolyMake and Lattice's LMK make utilities. He mentioned two problems that he was having with PolyMake—one relating to files existing in different directories and the other relating to memory usage.

I was unable to determine the exact nature of the directory problem that Mr. Holub mentioned. I can say, however, that we have literally thousands of users many of whom structure their projects hierarchically and apparently have no problem doing so with PolyMake. We use the same strategy internally at Polytron.

As regards Mr. Holub's comment about excessive memory usage, PolyMake has a feature that allows the user to determine how much memory it will consume. Use of this feature may have solved his problem.

Our customer support staff is very responsive to questions such as these that surface from time to time and would have been more than happy to help Mr. Holub resolve these "problems." Often all it takes is a simple phone call to resolve a perceived problem. When a genuine bug is discovered, we respond very quickly with a solution.

Incidentally, the current version of PolyMake is Version 2.1, and it contains many new and unique features in keeping with our market leadership position. These features include conditional constructs in both dependencies and operations, internal commands, shell control, some very powerful new macros, initialization operations, and the ability to determine time stamps of components of Polytron Version Control System (PVCS) archives and PolyLibrarian object module libraries. Also, Version 2.1 is about ten times faster and uses about 60 percent of the data space compared to PolyMake versions prior to Version 2.0.

Donald K. Kinzer
Polytron Corp.
1815 NW 169th Pl., Ste. 2110
Beaverton, OR 97006

Allen Holub replies:

I did call Polytron a year ago when I was having the directory problems discussed, and the staff couldn't come up with a solution then. Moreover, when I recently asked about the space-related problems, I was told that the new version (2.1) takes up even more memory than the version that I had. Mr. Kinzer's letter mentions that Version 2.1 uses 60 percent less data space, but he doesn't mention that it uses considerably more code space. Be that as it may, the added features may well make up for the amount of memory used. Polytron is sending me a copy of the new version, and I'll discuss it in a future C Chest.

DDJ

EVEN MORE POWER AND FLEXIBILITY

BRIEF 2.0

Users and industry press alike have unanimously proclaimed BRIEF as the best program editor available today. Now, the best gets better, with the release of BRIEF 2.0.

Straight from the box, BRIEF offers an exceptional range of features. Many users find that BRIEF is the only editor they'll ever need, with features like real, multi-level Undo, flexible windowing and unlimited file size. But BRIEF has tremendous hidden power in its exclusive macro language. With it, you can turn BRIEF into your own custom editor containing the commands and features you desire. It's fast and easy.

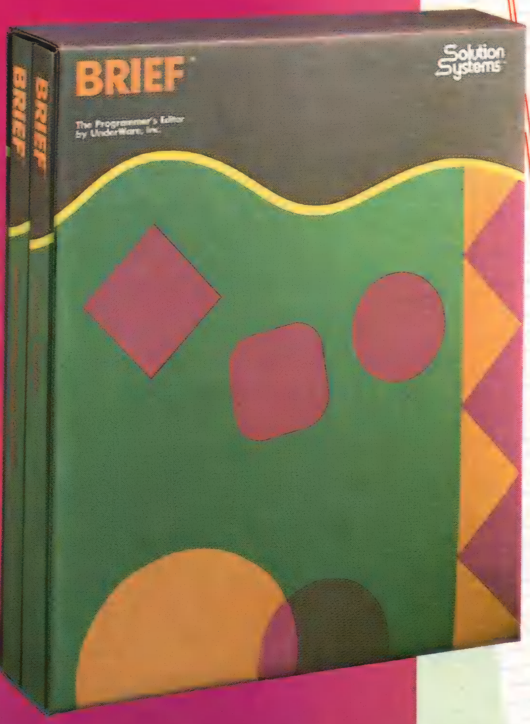
Jerry Pournelle, columnist for BYTE magazine summed it all up by saying BRIEF is, "**Recommended. If you need a general purpose PC programming editor, look no further.**" His point of view has been affirmed by rave reviews in C JOURNAL, COMPUTER LANGUAGE, DR. DOBB'S JOURNAL, DATA BASED ADVISOR, INFOWORLD AND PC MAGAZINE.

One user stated "BRIEF is one of the few pieces of software that I would dare call a masterpiece." Order BRIEF now and find out why. **BRIEF 2.0 is just $195. If you already own BRIEF, call for upgrade information.**

TO ORDER CALL: 1-800-821-2492 (in MA call 617-337-6963)

As always, BRIEF comes with a 30 day money-back satisfaction guarantee.

Solution Systems™
541 Main Street
Suite 410D
So. Weymouth, MA 02190
(617) 337-6963



Look at these BRIEF 2.0 enhancements!

Main Features:

- All new documentation with tutorials on basic editing, regular expressions **and** the BRIEF Macro Language.
- Setup program for easy installation and configuration. (Requires no knowledge of the macro language)
- Increased speed for sophisticated operations like Undo and Regular Expression Search.
- Expanded regular expressions, with matching over line boundaries.
- More block types, with marking by character, line or column.
- Command line editing (move cursor, add and delete characters, specify command parameters).
- Support for more programming languages.
- Optional borderless windows.
- Enhanced large display support, including wider displays.
- Reconfigurable indenting for C files (supports most indenting styles).

Plus the basic features that made BRIEF **SO** popular!

Basic Features:

- Full multi-level Undo
- Windows
- Edit many files at once
- File size limited only by disk space
- Automatic language sensitive indentation

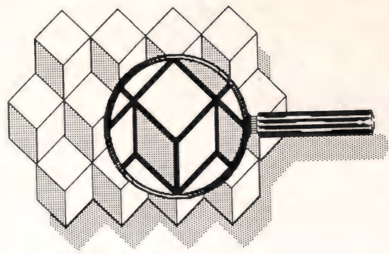
Requires an IBM PC or compatible with at least 192K RAM.

BRIEF is a trademark of UnderWare, Inc.

Solution Systems is a trademark of Solution Systems.

CIRCLE 142 ON READER SERVICE CARD

OF INTEREST



Languages

Language Processors has released a line of compilers for 386 machines running MS-DOS, Version 3.2. The languages currently available include LPI-COBOL, LPI-FORTRAN, LPI-RPG II, LPI-PL/I, LPI-PASCAL, and LPI-BASIC. Each product supports the 386 DOS-Extender from Phar Lap Software, which lets users run MS-DOS applications while facilitating access to the processing power of the 32-bit 386. Notable benefits include surmounting the 640K barrier, increasing overall performance, and enabling access of up to 4 Gbytes of memory.

Users who have developed applications using previous releases of LPI compilers will be able to transport those programs to 80386 machines running MS-DOS by recompiling them with the 386 version. Each compiler is priced at $695. Reader Service No. 16.

Language Processors Inc.
400-1 Totten Pond Rd.
Waltham, MA 02154
(617) 890-1155

JForth from **Delta Research** is based on the Forth-83 Standard and runs on the Commodore Amiga. Of particular interest is that high-level programs compile directly into machine code as opposed to interpreting tokens at run time. To cut development time, the compiler environment is interactive and allows incremental compiling. Utilities include a 68000 assembler and disassembler, search and sort routines, local variables, and floating point.

Amiga structures and constants are predefined in .j files, which correspond to the .h files used in C. Amiga library routines are called by name, and an object-oriented development environment is provided. Example programs demonstrate graphics, HAM mode, speech synthesis, and pull-down menus. Source code is furnished. JForth sells for $99.95. Reader Service No. 17.

Delta Research
201 D Street, Ste. 15
San Rafael, CA 94901
(415) 485-6867

Microware Systems Corp. is now shipping its OS-9/68020 C compiler for the 32-bit Motorola 68020 microprocessor. Based on the Kernighan & Ritchie standard, the compiler includes extensions for the OS-9 operating system (for compact disc-interactive new media technology). All compiler/assembler/linker options are controlled by an "intelligent executive" that governs compiler options and module-calling sequences.

Extensions to the OS-9 operating system environment include library functions for memory management and systems events as well as several library functions that provide compatibility with the proposed ANSI standard. The compiler uses the MC68881 math coprocessor for high-speed execution of complex math functions and can generate in-line floating-point instructions, link to MC68881 math libraries, or trap to a shared systemwide MC68881 math package. It can also take advantage of the MC68020's 32-bit arithmetic instructions and special addressing modes. The OS-9/68020 C compiler includes both MC68000 and MC68020 code generation packages and sells for $750. Reader Service No. 18.

Microware Systems Corp.
1900 N.W. 114th St.
Des Moines, IA 50322
(515) 224-1929

Oxxi's Benchmark Modula-2 for the Amiga implements the entire Modula-2 language as defined by Niklaus Wirth. Execution speed is enhanced because the compiler resides in memory. The compiler is activated directly from the editor by a hot key, so the time it would normally take to load an overlay from disk is eliminated. The editor contains more than 125 commands for dealing with multiple files, windows, and buffers. Demo programs include a freehand paint program, a desktop calculator, and a directory maintenance program. Programs written in Benchmark Modula-2 can be distributed without further licensing requirements from Oxxi.

Available add-on products include a C Language Standard Library, which allows programs written in C to be moved easily into the Modula-2 programming environment; Simplified Amiga Libraries, which includes functions for screen, window, sound, and device handling; Interchange File Format (IFF) Libraries; and Graphic Image Resource Management, consisting of both IFF libraries and the full documentation of the IFF format. The compiler sells for $199; the add-on products are available for $99 each. Reader Service No. 19.

Oxxi Inc.
1835-A Dawns Way
Fullerton, CA 92631
(714) 999-6710

Tools

Jasik Designs is now shipping MacNosy, Version 2, and The Debugger for the Mac II. For those of you not familiar with the product, MacNosy is a global, interactive decompiler that lets you recover the source code of any Macintosh application. Standard MacNosy features include on-line access to the Macintosh system structures and/or current values and to Macintosh trap names (system calls) and their parameter lists and allow disassembly of all 680x0 instructions (including the 69991 FPU and 68851 MMU).

Added features of this Mac II edition include new system structures to handle additions to ROM, including Color QuickDraw; the ability to disassemble the Mac II ROM; the identification of more than 600 internal procedures in ROM; and an increase in maximum text file size from 32K to 64K.

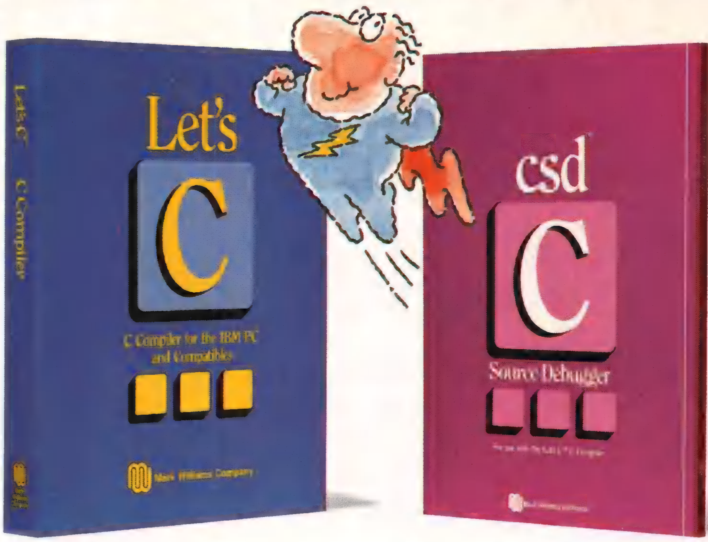
The Debugger monitors the execution of programs, allowing them to be arbitrarily stopped to trace execution. At this point you can view variables and structures as well as memory locations. The Mac II version includes the ability to run in single- or multi-screen mode and the addition of a dis-

PRESENTING THE DIFFERENCE BETWEEN FAST COMPILING AND FAST PROGRAMMING.

For compiling speed, you can't do better than Let's C. But to really speed up programming you can't do without the powerful source level debugger, *csd*.

If you want the power, portability and flexibility of C, start with the complete compiler, Let's C. For utilities, editor, compiling speed and fast, dense code, Let's C has it all.

But to get your programs up and running you need more. Because even the fastest compiler can't outrun bugs. You need the revolutionary C Source Debugger, *csd*.



LIMITED TIME OFFER FREE *csd* WITH LET'S C!

CUT DEVELOPMENT TIME IN HALF WITH *csd*

csd lets you bypass the time consuming frustrations of debugging–like long dumps and clunky assembler. With *csd*, you actually debug in C. You learn faster because you watch your program run in C. You finish faster because *csd* combines the speed of a compiler with the interactive advantages of an interpreter. The end result? Development time is sliced in half.

REVIEWERS ARE RAVING ABOUT LET'S C AND *csd*.

"Let's C is an inexpensive, high-quality programming package... with all the tools you will need to create applications."
–William G. Wong, *BYTE*, August 1986.

"The performance and documentation of the $75 Let's C compiler rival those of C compilers for the PC currently being sold for $500...highly recommended..."
–Marty Franz, *PC TECH JOURNAL*, August 1986.

"csd *is close to the ideal debugging environment...a definite aid to learning C and an indispensable tool for program development."*
–William G. Wong, *BYTE*, August 1986.

"This is a powerful and sophisticated debugger built on a well-designed, 'serious' compiler."
–Jonathon Sachs, *Micro/Systems Journal*, April, 1986

LET'S C AND *csd* FEATURES

NEW VERSION 4.0!

Let's C:

- Now compiles twice as fast
- Integrated edit-compile cycle: editor automatically points to errors
- Includes both small and large memory model
- Integrated environment or command line interface
- 8087 sensing and support
- Documentation features new lexicon format
- MS-DOS object compatible
- New make utility
- Fast compact code plus register variables
- Full Kernighan & Ritchie C and extensions
- Full UNIX compatibility and complete libraries
- Many powerful utilities including make, assembler, archiver, cc one-step compiling, egrep, pr, tail, wc
- MicroEMACS full screen editor with source included
- Supported by dozens of third party libraries
- For the IBM-PC and Compatibles
- Not copy protected

Sieve Benchmark

(Compile time in seconds)

Let's C: **2.8** (On 512K 6Mhz IBM-AT)
Turbo C: **3.89** (As advertised)

csd:

- Large and small memory model
- Debug in C source code, not assembler
- Monitor variables while tracing program
- Does not change program speed or size
- Provides separate source, evaluation, program and history windows
- On-line help screens
- Can interactively evaluate any C expression
- Can execute any C function in your program
- Trace back function
- Ability to set trace points
- Not copy protected

START TO FINISH, THERE'S NO BETTER ENVIRONMENT.

Get started with the right C compiler and you'll have everything you need for development–including source level debugging. On top of it all, Let's C and *csd* are today's best values in professional C programming tools. And most reliable: Mark Williams C compilers have been sold with DEC, Intel and Wang computers since 1981.

60 DAY MONEY BACK GUARANTEE

Mark Williams gives you a full 60 days to find out just how good Let's C and *csd* really are–or your money back.

So if you want more than a fast compiler–if you want your programs up and running fast, ask for Let's C and *csd*. You'll find them at your software dealer's, in the software department of your favorite bookstore, through the Express Program at over 5500 Radio Shacks or you can order now by calling **1-800-MWC-1700.***

*In Illinois call, 1-312-472-6659.

DDJ 1087

Mark Williams Company

1430 West Wrightwood, Chicago, Illinois 60614

© 1987 Mark Williams Company
Let's C is a registered trademark of the Mark Williams Company.
UNIX is a trademark of Bell Labs.

MARK WILLIAMS LET'S C AND *csd*. ONLY $75 EACH.

CIRCLE 102 ON READER SERVICE CARD

(continued from page 146)

play of floating-point registers.

The Mac II (universal) version of MacNosy and The Debugger are sold only in combination for $350. Versions for the Mac Plus and the Mac SE sell for $170. Reader Service No. 20.
Jasik Designs
343 Trenton Way
Menlo Park, CA 94024
(415) 322-1386

Solution Systems has introduced BRIEF 2.0, the latest version of its general-purpose programming editor for the PC. One especially nifty feature is an improved "undo" command, which allows programmers to reverse the effect of their last 300 commands. The update includes new documentation with tutorials on basic editing, regular expressions, and the internal macro language—a language that allows users to customize their editing environment to meet individual needs.

BRIEF 2.0 adds device drivers that support the Enhanced Graphics Adapter, the Hercules Graphics Plus Card, and the Wyse 700 display. It is now compatible with displays that have up to 127 lines by 255 characters. BRIEF 2.0 sells for $195; the upgrade cost for registered users is $60. Reader Service No. 21.
Solution Systems
335 Washington St.
Norwell, MA 02061
(617) 659-1571

MicroSolutions Computer Products has just released Version 2 of Uniform-PC, a utility that permits the use of more than 110 non-IBM PC disk formats on a standard PC/XT and more than 160 formats on an AT. Users of IBM PCs and compatibles can directly read, write, and initialize disks from most popular CP/M and MS-DOS computers. CP/M disks can be used just as though they were PC-DOS disks. New features include support for Apple CP/M, NorthStar CP/M hard-sector formats when used in conjunction with the new MatchPoint-PC, and MS-DOS formats from computers that do not use the IBM standard. Support is also provided for 48 TPI, 96 TPI, 8-inch, and 3.5-inch disk drives. Uniform-PC, Version 2, sells for $69.95. Reader Service No. 22.
MicroSolutions Computer Products
132 West Lincoln Highway
DeKalb, IL 60115
(815) 756-3411

Solutions International has just released SuperGlue, a graphics utility for the Macintosh that can be installed as a desk accessory and allows for font substitutions and text extraction from captured images. Images can be reduced or enlarged before they are printed to disk, and graphics can be automatically saved to a scrapbook file. The product has three parts: SuperImage Saver, a printer driver that prints text and graphics to disk as images; SuperViewer, a utility that allows users to open those images and print or copy them for use in other applications; and SuperViewerDA, a desk accessory version of SuperViewer. SuperGlue sells for $89.95. Reader Service No. 23.
Solutions International
29 Main St.
P.O. Box 989
Montpelier, VT 05602
(802) 229-9146

Hardware

ROMulator from **Grammar Engine** is, as its name implies, a ROM emulator. This hardware/software combination includes an in-circuit emulation module with associated cables and adapters along with the requisite host software. It can emulate standard (JEDEC) 24- and 28-pin ROMs, PROMs, and EPROMs; 8-, 16-, and 32-bit-word ROM modes; and supports daisy-chained modules of up to 8 ROMs. ROMulator software allows ROM software in Intel hex or Motorola S record format to be downloaded from any host via an RS-232 interface. When loaded, the software is then immediately available for access by the target system. Prices for the ROMulator (Model S256) start at $400. Reader Service No. 24.
Grammar Engine, Inc.
1021 Tipton Ct.
Westerville, OH 43081
(614) 882-6366

Applied Physics' BusMate PC/XT and BusMate AT cards offer hardware hackers or technicians effective tools for performing diagnosis on the PC bus system. Each card has a labeled gold pin for each signal line present on the bus and connections can be made to oscilloscopes, logic analyzers, or other test equipment that use standard test probes or microclips. Four light-emitting diodes monitor the system's power supply voltages. Each card also features a bus reset button that lets users reboot the system without powering down the machine. BusMate PC/XT sells for $79, and BusMate AT sells for $89. Reader Service No. 25.
Applied Physics Inc.
P.O. Box 2368
West Lafayette, IN 47906
(317) 497-1718

DDJ

MAKE NO MISTAKE...

"I made the mistake recently of using another mail order house and got bad service and old versions. Never Again."

William S. Gaut
Research Alternatives

We're Programmer's Connection, your best one-stop source for quality programmer's development tools for IBM personal computers and compatibles. Here are some important facts you should know about us and other dealers in our industry.

FREE Shipping. Shipping to U.S. customers is FREE via UPS Ground. If you want your order shipped via an express service, we'll only charge you the shipping carrier's standard rate with no special fees. Some dealers charge extra for shipping and then add rush charges for shipping via express services. Others may advertise "free" shipping but make up for it by charging extra handling fees.

Credit Cards. We'll charge your credit card only when we actually ship your order. Some dealers would charge your credit card at the time you place your order. This could leave you waiting for your shipment for weeks or months while they use your money interest-free.

Discounts. We discount all software products—even special order items. Every product in our advertised price list is shown with its list price and discounted price. We want you to know exactly how much you'll save on every product. We don't try to fool you by discounting some products and charging full retail for others.

FREE Buyer's Guide. Our new, 84-page Comprehensive Buyer's Guide & Catalog is the most extensive in the industry. It helps you find the best tools for the job by providing complete, functional descriptions for all of the software products in our regular product line.

Consistent Prices. We extend the same current prices to every customer regardless of where they see our ad. Some dealers vary prices in different ads and then ask you to mention which one you saw. This technique allows them to charge you the highest prices possible.

No Hidden Charges. The discount prices you see on the next two pages are all you pay. We don't charge extra for UPS Ground shipping, credit cards, COD orders, purchase orders, sales tax (except Ohio) or special handling (except for non-Canadian international orders).

Guarantees. We offer FREE 30-day no-risk return guarantees and 30-day evaluation periods on most of our products. Some dealers have no return options while others often charge restocking fees of 15% or more.

Quality Products. Our product line consists of hundreds of high quality software development tools specifically for IBM Personal Computers and compatibles. While some dealers try to carry every software product ever written, we carry only those that meet our very high standards for quality and value.

Latest Versions. The products we carry are the latest versions and come with the same manufacturer's technical support as if buying direct. While some dealers may participate in the software gray market, we're authorized to sell every product we carry.

Large Inventory. We have one of the largest inventories of programmer's development products in the industry. Most orders are shipped within 24 hours. And if we don't have a product in stock, we'll get it for you fast.

Meticulous Packaging. We'll give your shipment the extra protection needed to reach you in the best possible condition. First we'll protect your products from moisture by wrapping them in plastic. Then we'll insulate your box with high quality bubble-wrapping instead of the messy styrofoam chips that many other dealers use. Finally, we'll double-tape your box for extra strength.

Independence. Since we're not directly affiliated with any software publisher or developer, we can give you sound, unbiased advice. Unlike some dealers who have a special interest in promoting only certain products, we'll give you an objective look at the products we carry.

Noncommissioned Staff. Our courteous sales staff is always ready to help you. And if you aren't sure about your needs, our knowledgeable technical staff can give you sound, objective advice. Because they are noncommissioned, you won't be pressured into making a purchase.

As you can see, we're different from the other dealers in our industry. Our customers keep coming back because we consistently provide the highest quality service and the lowest prices. So call us today and experience these differences for yourself.

Turn the page for our product list and ordering information.

PROGRAMMER'S
QUALITY INTEGRITY
Service
SUPPORT PRICE
CONNECTION

...MAKE THE CONNECTION

CIRCLE 129 ON READER SERVICE CARD

ai - expert systems

	List	Ours
1st-CLASS *by Programs in Motion*	495	399
EXSYS *Development Software by EXSYS*	395	309
EXSYS Runtime System	600	469
LEVEL5 *by Information Builders* ... *New*	685	569
Logic-Line Series *All varieties by Thunderstone*	CALL	CALL

ai - lisp language

	List	Ours
GCLISP *Golden Common LISP by Gold Hill*	495	CALL
GCLISP 286 Developer *by Gold Hill*	1190	CALL
Microsoft LISP *Common LISP*	250	149
QNIAL *Combines LISP & APL by NIAL Systems*	CALL	CALL
Sapiens MAKE & V8 ... *New*	439	379
Sapiens MAKE ... *New*	179	155
Sapiens V8 *Virtual Memory Manager* ... *New*	300	265
Star Sapphire LISP *Compiler by Sapiens* ... *New*	495	429
TransLISP PLUS *from Solution Systems*	195	125

ai - prolog language

	List	Ours
Arity Combination Package	1095	979
Expert System Development Pkg	295	229
File Interchange Toolkit	50	44
PROLOG Compiler & Interpreter	650	569
Screen Design Toolkit	50	44
SQL Development Package	295	229
Arity PROLOG Interpreter	295	229
Arity Standard Prolog	95	77
LPA microPROLOG *All Varieties*	CALL	CALL
MPROLOG Language Primer *LOGICWARE*	50	45
MPROLOG P500 *by LOGICWARE*	495	395
MPROLOG P550 *w/Primer by LOGICWARE*	220	175
Turbo PROLOG *by Borland Intl*	100	64
Turbo PROLOG Toolbox *by Borland Intl*	100	64

ai - smalltalk language

	List	Ours
Smalltalk/V	100	84
EGA/VGA Color Option	50	45
Goodies Diskette	50	45
Smalltalk/Comm	50	45

ai - texas instruments

	List	Ours
Arborist *Decision Tree Software*	595	519
PC Scheme *Lisp*	95	84
Personal Consultant Easy	495	435
Personal Consultant Image	495	435
Personal Consultant Online	995	869
Personal Consultant Plus	2950	2589
Personal Consultant Runtime	95	84

ada language

	List	Ours
AdaVantage *GSA-Validated by Meridian Software*	795	735
AdaVantage Utility Packages ... *New*	50	47
DOS Environment Package ... *New*	50	47
Janus/ADA C Pak *by R&R Software*	95	84
Janus/ADA D Pak *by R&R Software*	1250	1059
Janus/ADA ED Pak *by R&R Software*	395	349

apl language

	List	Ours
APL*PLUS PC *by STSC*	595	424
APL*PLUS PC Spreadsheet Mgr *by STSC*	195	139
APL*PLUS PC Tools Vol 1 *by STSC*	295	199
APL*PLUS PC Tools Vol 2 *by STSC*	85	58
APL*PLUS PS/2 *by STSC* ... *New*	595	424
ATLAS*GRAPHICS *by STSC*	450	329
Financial/Statistical Library *by STSC*	275	189
Pocket APL *by STSC*	95	69
STATGRAPHICS *by STSC*	895	649

assembly language

	List	Ours
386 ASM/LINK *Cross Asm by Phar Lap*	495	389
8088 Assembler *w/Z-80 Translator by 2500 AD*	100	89
ASMLIB *Function Library by BC Assoc*	149	125
asmTREE *B-Tree Dev System by BC Assoc*	395	329
Cross Assemblers *Various by 2500 AD*	CALL	CALL
EZASM *by C Source* ... *New*	70	59
Microsoft Macro Assembler ... *New Version*	150	93
Turbo Debugger *by Speedware* ... *New*	89	79
Turbo Editasm *by Speedware*	99	84
Visible Computer: 8088 *by Software Masters*	80	64

basic language

	List	Ours
87 Software Pak *by Hauppauge*	180	149
db/Lib *for QuickBASIC by AJS Publishing* ... *New*	139	119
Finally *by Komputerwerk*	99	85
MACH 2 *by Micro Help*	69	55
Microsoft QuickBASIC ... *$20 Rebate Until 10/15*	99	63
QBase *Relational Database by Crescent*	89	79
Quick-Tools *by BC Associates*	130	109
QuickPak *by Crescent Software*	69	59
Scientific Subroutine Library *by Peerless*	125	99
Screen Sculptor *by Software Bottling*	125	91
Stay-Res *by MicroHelp*	69	55
True Basic *w/Run-time* ... *Special Price*	200	99
True Basic	100	79
Run-time Module	100	79
Various Utilities	50	41
Turbo BASIC Compiler *by Borland Intl*	100	64

blaise products

	List	Ours
ASYNCH MANAGER *Specify C or Pascal*	175	135
C TOOLS PLUS/5.0 ... *New Version*	129	99
KeyPlayer *Super Batch Program* ... *New*	50	45
LIGHT TOOLS *for Datalight C*	100	65
PASCAL TOOLS	125	95
PASCAL TOOLS 2	100	79
PASCAL TOOLS & TOOLS 2	175	135
RUNOFF *Text Formatter*	50	45
TURBO ASYNCH PLUS	100	79
TURBO C TOOLS	129	99
TURBO POWER TOOLS PLUS	100	79
VIEW MANAGER *Specify C or Pascal*	275	199

borland products

	List	Ours
EUREKA *Equation Solver*	167	105
Reflex: The Analyst	195	128
Sidekick	85	57
Superkey	100	64
Turbo Basic Compiler	100	64
Turbo Basic Database Toolbox ... *New*	100	64
Turbo Basic Editor Toolbox ... *New*	100	64
Turbo Basic Telecom Toolbox ... *New*	100	64
Turbo C Compiler *(Call for support products)*	100	64
Turbo Lightning and Word Wizard	150	94
Turbo Lightning	100	64
Turbo Lightning Word Wizard	70	47
Turbo Pascal and Tutor	125	85
Turbo Pascal	100	64
Turbo Pascal Tutor	40	24
Turbo Pascal Database Toolbox	70	41
Turbo Pascal Editor Toolbox	70	41
Turbo Pascal Gameworks Toolbox	70	41
Turbo Pascal Graphix Toolbox	70	41
Turbo Pascal Jumbo Pack	300	219
Turbo Pascal Numerical Methods Toolbox	100	64
Turbo Prolog Compiler	100	64
Turbo Prolog Toolbox	100	64

c compilers

	List	Ours
C86PLUS *by Computer Innovations*	497	375
DeSmet C *w/Debugger & Large case*	209	184
DeSmet C *w/Debugger only*	159	138
Eco-C *Complete System by Ecosoft*	140	119
Lattice C Compiler *vers. 3.2 from Lattice*	500	265
Mark Williams Let's C w/csd	75	54
Microsoft C Compiler w/CodeView ... *New version*	450	269
Microsoft QuickC Compiler ... *New*	99	63
Optimum-C *by Datalight*	139	95
Turbo C Compiler *by Borland*	100	64
Uniware 68000/10/20 Cross Compiler *by SDS*	995	829

c interpreters

	List	Ours
C-terp *by Gimpel, Specify compiler* ... *New Version*	300	219
C Trainer *with Book by Catalytix*	122	87
Instant C *by Rational Systems* ... *New Version*	500	369
Introducing C *by Computer Innovations*	125	99
Run/C *by Age of Reason*	120	69
Run/C Professional *by Age of Reason*	250	145

c utilities

	List	Ours
Blackstar C Library *by Sterling Castle* ... *New*	99	78
C++ *by Guidelines w/version 1.1 kernel*	195	172
c-tree & r-tree Combo *by FairCom*	650	519
c-tree *ISAM File Manager*	395	315
r-tree *Report Generator*	295	239
Csharp Realtime Toolkit *by Systems Guild* ... *New*	600	539
Curses Window Dev Pkg *by Aspen Scientific* ... *New*	119	105
dBx *dBASE to C Translator by Desktop AI*	350	299
with Source Code ... *New*	550	419
Flash-up Windows *by Software Bottling*	90	78
GraphiC *Color version by Sci Endeavors*	350	274
GRAFLIB *by Sutrasoft*	175	159
HALO *Graphics by Media Cybernetics*	300	205
HALO Development Pkg *for Microsoft*	595	389
The HAMMER *by OES Systems*	195	129
PANEL *Forms Management by Roundhill*	295	215
PANEL/TC *for Turbo C by Roundhill* ... *New*	129	95
PANEL Plus *by Roundhill*	495	395
PC Lint *by Gimpel Software*	139	99
PHACT *from UniPress* ... *New*	249	229
PLOTHP *by Sutrasoft*	175	159
RTC PLUS *Fortran to C by Cobalt Blue*	450	399
Scientific Subroutine Library *by Peerless*	175	135
TE Text Editor *source by Sub Systems* ... *New*	95	85
Vitamin C *by Creative Programming*	225	158
VC Screen *Forms Designer*	100	79
Zview *by Data Mgmt Consultants*	245	139

cobol language

	List	Ours
COBOLspII *by Flexus*	395	329
FPLIB *for Realia COBOL by BC Associates*	149	129
Micro Focus COBOL *See Micro Focus Section*		
Microsoft COBOL *See Microsoft Section*		
PCDT *by Pro-Code* ... *New*	995	895
Realia COBOL with RealMENU ... *New Version*	1145	899
Realia COBOL ... *New Version*	995	783
RealCICS	995	783
RM/COBOL *by Ryan-McFarland*	950	CALL
RM/COBOL 85 *by Ryan-McFarland*	1250	CALL
RM/NET+5 *by Ryan-McFarland* ... *New*	300	CALL
RM/Screens ... *New*	395	CALL
SCREENIO *by Norcom*	400	379
screenplay *for COBOL by Flexus*	175	129

css products

	List	Ours
Combo Package *by Custom Software Systems*	199	175
PC/SPELL *Spelling Checker*	49	45
PC/TOOLS *UNIX-like Utilities*	49	45
PC/VI *vi Editor*	149	99

debuggers & profilers

	List	Ours
386 DEBUG *Cross Debugger by Phar Lap*	195	129
Advanced Trace-86 *by Morgan Computing*	175	115
Codesmith-86 *by Visual Age*	145	98
DSD87 *by Soft Advances*	125	79
MiniProbe *by Atron*	395	369
Periscope I *with Board by Periscope*	345	275
Periscope II *with NMI Breakout Switch*	175	139
Periscope II-X *Software only*	145	105
Periscope III *8 MHz version*	995	795
Periscope III *10 MHz version*	1095	899
The PROFILER *with Source Code by DWB*	125	89
TURBOsmith *Source debugger for Turbo Pascal*	69	59
The WATCHER *Profiler by Stony Brook*	60	51

disk utilities

	List	Ours
Back-It *by Gazelle Systems*	100	89
Disk Optimizer *by Softlogic Systems*	60	55
Disk Technician *by Prime Solutions* ... *New*	100	89
FASTBACK *by 5th Generation Systems*	179	129
Vcache *by Golden Bow Systems* ... *New*	50	47
Vopt *by Golden Bow Systems* ... *New*	50	47
Vfeature *by Golden Bow Systems* ... *New*	80	74
Vfeature Deluxe *by Golden Bow Systems* ... *New*	120	111
XenoCopy-PC *by XenoSoft*	80	69

dos utilities

	List	Ours
Advanced Norton Utilities	150	99
Command Plus *by ESP Software*	80	69
FANSI-CONSOLE *by Hersey Micro*	75	62
Mace Utilities *Paul Mace Software* ... *New*	99	89
MicroHelp Utility *by MicroHelp*	59	49
Norton Commander *by Peter Norton*	75	55
Norton Utilities *by Peter Norton*	100	59
OPAL *Shell Language by Software Factory*	99	89
Q-DOS II *by Gazelle Systems*	70	59
Taskview *by Sunny Hill Software*	80	55

essential products

	List	Ours
C Utility Library	185	119
Essential Comm Library *with Debugger*	250	189
Essential Comm Library *Software Only*	185	125
Breakout Debugger Only *Any language*	125	89
Essential Graphics	250	183

forth language

	List	Ours
CFORTH *Native Code Compiler by LMI*	300	229
FORTH/83 Metacompiler *Specify Target*	750	599
PC/FORTH *by Laboratory Microsystems*	150	109
PC/FORTH+ *by Laboratory Microsystems*	250	199
Programmer's Package #1 *by LMI* ... *New*	250	199
Programmer's Package #2 *by LMI* ... *New*	350	279
Programmer's Package #3 *by LMI* ... *New*	500	399
UR/FORTH *Also Available for OS/2 by LMI*	350	279
UR/FORTH Libraries	500	395

fortran language

	List	Ours
50 MORE: FORTRAN *by Peerless Scientific*	125	95
ACS Time Series *by Alpha Computer Service*	495	389
AUTOMATED PROGRAMMER *by KGK Automated* *New*	250	183
Essential Graphics *by Essential Software*	250	183
For-Winds *by Alpha Computer Service*	90	69
Forlib-Plus *by Alpha Computer Service*	70	44
FORTLIB *by Sutrasoft*	125	109
FORTRAN Addendum *by Impulse Engr*	95	85
FORTRAN Addenda *by Impulse Engr*	165	138
GRAFLIB *by Sutrasoft*	175	159
HALO *Graphics by Media Cybernetics*	300	205
I/O PRO *w/No Limit Library by MEF*	250	219
Microcompatibles Combo Package	240	215
Grafmatic	135	117
Plotmatic	135	117
Microsoft FORTRAN *w/CodeView*	450	269
No Limit Library *by MEF Environmental*	129	109
Numerical Analyst *by MAGUS*	295	249
PANEL *by Roundhill Computer Systems*	295	215
PLOTHP *by Sutrasoft*	175	159
RM/FORTRAN *by Ryan-McFarland* ... *New Version*	595	CALL
RTC PLUS *Fortran to C by Cobalt Blue*	450	399
Scientific Subroutine Lib *by Peerless*	175	135
Statistician *by Alpha Computer Service*	295	235
STATLIB.GL: *by Peerless* ... *New Version*	295	239
STATLIB.TSF: *by Peerless* ... *New Version*	295	239
Strings & Things *by Alpha Computer Service*	70	45

greenleaf products

	List	Ours
Greenleaf C Sampler *for Turbo C & QuickC* ... *New*	95	69
Greenleaf Comm Library	185	125
Greenleaf Data Windows Library	225	155
with Source Code	395	249
Greenleaf Functions	185	125

help utilities

	List	Ours
HELP/Control *by MDS*	125	99
On-line Help *from Opt-Tech*	149	99
SoftScreen/HELP *by Dialectic Systems*	195	149

lattice products

	List	Ours
Lattice C Compiler *ver 3.2 from Lattice*	500	265
with Library Source Code	900	495
C Cross Reference Generator	50	37
with Source Code	200	139
C-Food Smorgasbord *Function Library*	150	95
with Source Code	300	179
C-Sprite *Source Level Debugger*	175	119
Curses *Screen Manager*	125	85
with Source Code	250	169
dBC *Specify dBC II or dBC III*	250	169
with Source Code	500	356
dBC III Plus	750	594
with Source Code	1500	1184
LMK *Make Facility*	195	138
RPG II Combo *All three items below*	1100	875
RPG II Compiler *No Royalties*	750	625
SEU *Source Entry Utility*	250	199
Sort/Merge	250	199
Screen Design Aid Utility *for RPG II*	350	309
SecretDisk *File Encryption Utility*	120	88
SideTalk *Resident Communications*	120	88
SSP/PC *Scientific Subroutine Library*	350	269
Text Management Utilities	120	88
TopView Toolbasket *Function Library*	250	178
with Source Code	500	356

metagraphics products

	List	Ours
LightWINDOW/C *for Datalight C*	95	79
FontWINDOW	95	79
FontWINDOW/PLUS	275	229
MetaWINDOW *No Royalties*	195	159
MetaWINDOW/PLUS	275	229
TurboWINDOW/C *for Turbo C*	95	79
TurboWINDOW/Pascal *for Turbo Pascal*	95	79

micro focus products

	List	Ours
Micro Focus Level II COBOL *w/Animator*	495	395
Level II COBOL	349	279
Level II Animator	195	155
Micro Focus Level II COBOL/ET for UNIX	CALL	CALL
Micro Focus Personal COBOL	149	119
Micro Focus Professional COBOL	2000	1595
Micro Focus VS COBOL/XENIX	1495	1195
Micro Focus Support Products:		
COBOL/IQ *Ad hoc Report Writer*	495	395

CIRCLE 129 ON READER SERVICE CARD

FORMS-2	295	235
SOURCEWRITER	995	795

microport products

386 Unlimited License Kit	249	209
AT Unlimited License Kit	249	209
DOSMerge286 *Run DOS and UNIX together*	149	125
DOSMerge386 *Run DOS and UNIX together*	CALL	CALL
System V/386 Combination	799	699
386 Runtime System	199	169
386 Software Development System	499	429
Text Preparation System	199	169
System V/AT Combination	549	465
AT Runtime System	199	169
AT Software Development System	249	209
Text Preparation System	199	169

microsoft products

Microsoft BASIC Compiler *for XENIX*	695	419
Microsoft BASIC Interpreter *for XENIX*	350	209
Microsoft C Compiler *w/CodeView* .... *New Version*	450	269
Microsoft COBOL Compiler *with COBOL Tools*	700	429
for XENIX	995	609
Microsoft FORTRAN *Optimizing Compiler/CodeView*	450	269
Microsoft FORTRAN *for XENIX*	695	419
Microsoft Learning DOS	50	36
Microsoft LISP *Common LISP*	250	149
Microsoft MACH 10 *with Mouse & Windows*	549	369
Microsoft MACH 10 *Board only*	399	279
Microsoft Macro Assembler .... *New Version*	150	93
Microsoft Mouse *for IBM PS/2* .... *New*	175	114
Microsoft Mouse *Bus Version*	175	114
Microsoft Mouse *Serial Version*	195	124
Microsoft muMath *Includes muSIMP*	300	179
Microsoft Pascal Compiler	300	179
for XENIX	695	419
Microsoft QuickBASIC .... *$20 Rebate Until 10/15*	99	63
Microsoft QuickC .... *New*	99	63
Microsoft Sort	195	125
Microsoft Windows	99	63
Microsoft Windows Development Kit	500	299
Microsoft Word	450	269

mks products

MKS AWK *Includes Book Offer until 10/30* ... *New*	75	65
MKS Toolkit *with MKSVI Editor*	139	114
MKSVI *Editor by MKS* .... *New*	75	65

modula-2 language

EXE2LNK *MASM Interface by PMI*	49	45
LOGITECH Modula-2 Development System.... *New*	249	199
Modula-2 Compiler Pack .... *New*	99	79
Modula-2 Toolkit .... *New*	169	139
LOGITECH Modula-2 ROM Pkg & X RT Debugger	299	239
LOGITECH Modula-2 Window Pkg	49	39
Macro2 *Macro preprocessor by PMI*	89	79
ModBase *by PMI*	89	79
ModGraph *by TEQNA* .... *New*	50	45
Repertoire *by PMI*	89	75
Science & Engrg Tools *by Quinn-Curtis*	75	67
Universal Graphics Library *by Quinn-Curtis*	150	119

mouse products

LOGIMOUSE BUS *with PLUS Pkg by LOGITECH*	119	98
with PLUS & PC Paintbrush	149	119
with PLUS & CAD Software	189	153
with PLUS & CAD & Paint	219	179
with First Publisher .... *New*	CALL	CALL
LOGIMOUSE C7 *with PLUS Pkg, Specify Connector*	119	98
with PLUS & PC Paintbrush	149	119
with PLUS & CAD Software	189	153
with PLUS & CAD & Paint	219	179
with First Publisher .... *New*	CALL	CALL
Microsoft Mouse *See Microsoft Section*		

other languages

ACTOR *by Whitewater Group*	495	419
CCS MUMPS *Single-User by MGlobal*	60	50
CCS MUMPS *Single-User Multi-Tasking*	150	129
CCS MUMPS *Multi-User*	450	359
Marshal Pascal *by Marshal Language Systems*	189	165
Pascal-2 *by Oregon Software*	395	325
Personal REXX *by Mansfield Software*	125	99
SNOBOL4+ *by Catspaw*	95	80

other products

Carbon Copy *by Meridian Technology* .... *New*	195	159
Dan Bricklin's Demo Pgm *by Software Garden*	75	57
Dan Bricklin's Demo Tutorial	50	45
Fast Forward *by Mark Williams*	70	59
Instant Replay *by Nostradamus*	150	CALL
MicroTEX *Typesetting from Addison-Wesley*	295	CALL
Net-Tools *by BC Associates*	149	129
Norton Guides *Specify Language* .... *New*	100	75
OPT-Tech Sort *by Opt-Tech Data Proc*	149	99
PC/TOOLS *by Custom Software*	49	45
Screen Machine *by MicroHelp*	79	59
SuperSort *by LifeStyle* .... *New*	139	119

phoenix products

Pasm86 *Macro Assembler version 2.0*	195	108
Pdisk *Hard Disk & Backup Utility*	145	99
Pfantasy Pac *Phoenix Combo*	995	595
Pfinish *Execution Profiler*	395	209
Pfix86plus *Symbolic Debugger*	395	209
PforCe *Specify C Compiler*	395	209
PforCe++ *Specify C Compiler and C++*	395	209
Plink86plus *Overlay Linker*	495	275
Pmaker *Make Utility*	125	78
Pmate *Macro Text Editor*	195	108
Pre-C *Lint Utility*	295	154
Ptel *Binary File Transfer Program*	195	108

polytron products

PolyBoost *The Software Accelerator*	80	64
PolyDesk III	99	72
PolyDesk III Archivist	50	42
PolyDesk III Cryptographer	50	42
PolyDesk III Talk	70	52
PolyLibrarian *Library Manager*	99	89
PolyLibrarian II *Library Manager*	149	129
PolyMake *UNIX-like Make Facility*	149	129
PolyShell	149	129
Polytron C Beautifier	50	45
PolyXREF *Complete Cross Ref Utility*	219	185
PolyXREF *One language only*	129	109
PVCS Corporate *Version Control System*	395	329
PVCS Personal	149	129

program mgmt utilities

Interactive EASYFLOW *by Haventree*	150	125
PrintQ *by Software Directions*	89	84
Quilt Computing Combo Package	250	199
QMake *Program Rebuild Utility*	99	79
SRMS *Software Revision Mgmt System*	185	159
Source Print *by Aldebaran Labs*	97	75
TLIB *Version Control System by Burton*	100	89
Tree Diagrammer *by Aldebaran Labs*	77	67

raima products

dbQUERY *Single-User Query Utility*	195	129
Single-User with Source Code	495	389
Multi-User	495	389
Multi-User with Source Code	990	699
dbVISTA *Single-User DBMS*	195	129
Single-User with Source Code	495	389
Multi-User	495	389
Multi-User with Source Code	990	699

sco products

Complete XENIX System V *by SCO*	1295	994
Development System	595	499
Operating System *Specify XT or AT*	595	499
Text Processing Package	195	144
Lyrix *by SCO*	595	449
SCO Professional *1-2-3 Workalike for XENIX*	795	595
SCO XENIX-NET	595	495
XENIX System V 386 *by SCO* .... *New*	CALL	CALL

softcraft products

Btrieve *ISAM Mgr with No Royalties*	245	184
Xtrieve *Query Utility*	245	184
Report Option *for Xtrieve*	145	99
Btrieve/N *for Networks*	595	454
Xtrieve/N	595	454
Report Option/N *for Xtrieve/N*	345	269

text editors

Brief & dBrief Combo *from Solution Systems*	275	CALL
Brief	195	CALL
dBrief *Customizes Brief for dBASE III*	95	CALL
de *by David Livshin* .... *New*	75	65
Epsilon *Emacs-like editor by Lugaru*	195	147
KEDIT *by Mansfield Software*	125	98
Micro/SPF *by Phaser Systems*	175	139
Microsoft Word	450	269
PC/VI *by Custom Software Systems*	149	99
SPF/PC *by Command Technology Corp*	CALL	CALL
Vedit Plus *by CompuView*	185	128

turbo pascal utilities

ALICE *Interpreter by Software Channels*	95	66
DOS/BIOS & Mouse Tools *by Quinn-Curtis*	75	67
Flash-up Windows *by Software Bottling*	90	78
MACH 2 *for Turbo Pascal by Micro Help*	69	55
MetraByte D/A Tools *by Quinn-Curtis*	100	89
Science & Engrg Tools *by Quinn-Curtis*	75	67
Screen Sculptor *by Software Bottling*	125	91
Speed Screen *by Software Bottling*	35	32
System Builder *by Royal American*	150	129
IMPEX *Query Utility*	100	89
Report Builder	130	115
TDebugPLUS *by TurboPower Software*	60	49
Tmark *by Tangent Systems* .... *New*	80	69
Turbo EXTENDER *by TurboPower Software*	85	64
Turbo OPTIMIZER *by TurboPower*	75	65
with Source Code	125	98
Turbo Professional *by Sunny Hill*	70	45
TurboHALO *from IMSI*	129	98
TurboPower Utilities *by TurboPower*	95	78
TurboRef *by Gracon Services*	50	35
TURBOsmith *Source Debugger by Visual Age*	69	59
Universal Graphics Library *by Quinn-Curtis*	150	119

wendin products

Operating System Toolbox	99	79
PCNX *Operating system*	99	79
PCVMS *Similar to VAX/VMS*	99	79
Wendin-DOS *Multitasking DOS*	99	85
Wendin-DOS Application Developer's Kit ..... *New*	99	85
XTC *Text Editor w/Pascal source*	99	75

xenix/unix products

Btrieve *ISAM File Mgr by SoftCraft*	595	454
C-terp *by Gimpel, Specify compiler*	498	379
c-tree *ISAM Mgr by FairCom*	395	315
dBx *with Library Source by Desktop AI*	550	419
Desqview *from Quarterdeck* .... *New*	100	85
DIRECTORY SHELL *by American Mgmt Systems* *New*	349	295
DOSIX *Console Version by Data Basics*	399	349
DOSIX *User Version by Data Basics*	199	179
Micro Focus Products *See Micro Focus Section*		
Microport Products *See Microport Section*		
Microsoft Products *See Microsoft Section*		
PANEL Plus *by Roundhill Computer Systems*	795	535
REAL-TOOLS *Binary Version by PCT*	149	89
Library Source Version	399	289
Complete Source Version	999	729
RM/COBOL *by Ryan-McFarland*	1250	CALL
RM/FORTRAN *by Ryan-McFarland*	750	CALL
SCO Products *See SCO Section*		

Call or write for our FREE Comprehensive Buyers Guide
Terms subject to change.

®Copyright 1987 Programmer's Connection Incorporated.

LOWEST PRICES
Due to printing lead times, some of our current prices may differ from those shown here. Call for latest pricing.

FREE SHIPPING
Orders within the USA (including Alaska & Hawaii) are shipped FREE via UPS. Express shipping is available at the shipping carrier's standard rate with no rush fees or handling charges. To avoid delays when ordering by mail, please call first to determine the exact cost of express shipping.

CREDIT CARDS
VISA and MasterCard are accepted at no extra cost. Your card is charged when your order is shipped. Mail orders please include credit card expiration date and authorized signature.

CODs AND POs
CODs and Purchase Orders are accepted at no extra cost. No personal checks are accepted on COD orders. POs with net 30-day terms (with initial minimum order of $100) are available to qualified US accounts only.

SALES TAX
Orders outside of Ohio are not charged state sales tax. Ohio customers please add 6% Ohio tax or provide proof of tax-exemption.

INTERNATIONAL ORDERS
Shipping charges for International and Canadian orders are based on the shipping carrier's standard rate. Since rates vary between carriers, please call or write for the exact cost. International orders (except Canada), please include an additional $10 for export preparation. All payments must be made with US funds drawn on a US bank. Please include your telephone number when ordering by mail. Due to government regulations, we cannot ship to all countries.

VOLUME ORDERS
Volume orders may qualify for additional discounts. Call us for special pricing.

SOUND ADVICE
Our knowledgeable technical staff can answer technical questions, assist in comparing products and send you detailed product information tailored to your needs.

30-DAY GUARANTEE
Most of our products (excluding books) come with a 30-day documentation evaluation period or a 30-day return guarantee. Please note that some manufacturers restrict us from offering guarantees on their products. Call for more information.

MAIL ORDERS
Please include your telephone number on all mail orders. Be sure to specify computer, operating system and any applicable compiler or hardware interface(s). Send mail orders to:

Programmer's Connection
Order Processing Department
136 Sunnyside Street
Hartville, OH 44632

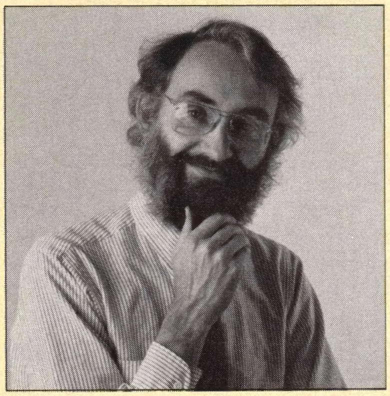
USA	**800-336-1166**
CANADA	**800-225-1166**
OHIO & ALASKA (Collect)	**216-877-3781**
TELEX	**9102406879**
EASYLINK	**62806530**
INTERNATIONAL	**216-877-3781**
CUSTOMER SERVICE	**216-877-1110**

Hours: Weekdays 8:30 AM to 8:00 PM EST.

PROGRAMMER'S Service CONNECTION
QUALITY INTEGRITY SUPPORT PRICE

CIRCLE 129 ON READER SERVICE CARD

SWAINE'S FLAMES



Doesn't your heart just go out to Lotus Development Corporation? First everybody and his brother ripped off the Lotus 1-2-3 user interface. Then somebody hit on the idea of compiling 1-2-3 spreadsheets into objects that can be manipulated without 1-2-3, so that one copy of 1-2-3 may be all a company needs. And now the latest wrinkle in the spreadsheet counterpane: Microsoft has slipped in with a PC spreadsheet product that reads 1-2-3 files and converts 1-2-3 macros.

It reminds me of a story I wrote once.

Back when I worked for *InfoWorld* and VisiCalc and SuperCalc were in flower, I wrote a regular back-page column that posed mathematical puzzles in the format of mystery stories featuring a character named Usasi and often touching obliquely on computer industry issues. In 1983, I wrote something like this:

I was sitting at my favorite table in Mister Bob's on Polk Street, explaining a fine forensic point to my associate Casey Standard and smart-mouthed lawyer Bette Noire. . . . Casey asked if we had heard of a programming case Mr. Usasi had in which a PROLOG expert system had been ported to a machine on which the values true and false had the opposite representation from what the programmer expected. Every true became a false, every full became an empty. Truth values were reversed universally. It was a real mess. Mr. Usasi was called in as an expert on expertise to advise them.

"Simple," I said. "Just reverse the interpretation of the output. You know, there was a clever database program on micros back in the 70s that looked a lot like expert systems of today, but was much simpler. You could tell it, 'Bartholomew's maiden aunt's Abigail,' and 'Bartholomew's birthday isn't February 29,' and ask questions like 'What's Bartholomew's favorite color?' It used Soundex coding. What was it called?

"Anyway, this all reminds me of that time in Guadalajara when I broke up a software smuggling ring that could have torpedoed the electronic spreadsheet market."

"That doesn't make any sense," Bette objected. "You don't have to smuggle software; you can just—"

"I'd insinuated myself into the smuggling ring," I went on, "a group of Orange County numerologists who had got it into their heads that balance sheets belonged to the masses and who were smuggling in spreadsheets to undermine the market leaders, which were Sorcim and VisiCorp back then.

"For deep and subtle numerological reasons, these Orange County spreadsheet smugglers had to divide the gold dust in which they had been paid into equal shares, with each smuggler getting exactly as many shares as there were smugglers who got shares, and with no undistributed shares.

"Now I had infiltrated the band and had studied a little numerology to beat them at their own game. I siezed on the fact that the smugglers secretly didn't want an even distribution of the spoils and suggested a scheme that would result in the same number of shares being distributed—which mattered to their numerological principles—and that would still allow an uneven distribution of shares—which appealed to the cupidity of the more powerful members.

"Give me exactly one share for each smuggler, I told them, counting me as one of the smugglers of course, and give each smuggler a different even number of shares, assigning them anywhichway. I gave them solid numerological reasons for the plan, and they bought it. I also stipulated, since I had demonstrated my value by coming up with the plan, that nobody get as much as twice my allotment, and they bought that, too."

Casey frowned at me. "I'm afraid this is a little far-fetched, Mickey."

"Sure it was," I explained patiently. "I was just playing along with them. I'm not into that numerological stuff. Anyway, after they solemnly agreed to the plan and swore their numerological oaths, they started divvying up the gold dust and discovered what I had pulled on them—but by then it was too late. Athough they didn't much like it, they had the integrity to stand by their numerological principles." I signaled for the check.

"That's the end?" Casey asked. "But what happened? What had you pulled on them?"

"Do the math," I said. "It's all in the math."

You can surely solve Mickey's little puzzle, but what about the upside-down expert system Casey mentioned? Because of its closed-world implementation, PROLOG does not treat true and false complementarily. What would happen if the base truth values in a PROLOG program were reversed? And what was the name of that database program?

Michael Swaine

Michael Swaine
editor-in-chief

How A C Programmer Became A Screen Star

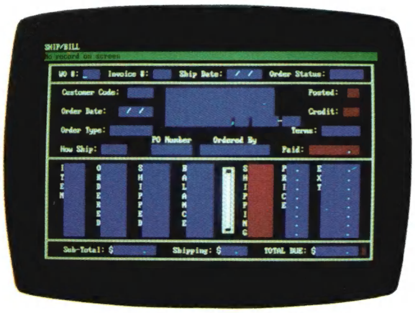
Screens, the Visible Part of Your Program.

A program is often judged by how well the screens are executed. However, the real creativity lies in what goes on behind the screens.

ScreenStar is a product that allows your real creativity to light up the screen. It reduces costly screen, window, and data validation development time.

You Take the Bows, We Write the Code.

Our natural drawing commands allow you to paint any screen imaginable. Press one key when you are satisfied and ScreenStar produces concise, commented, ready-to-compile code. This allows immediate testing of the I/O screens, including smooth, even scrolling between multiple screens.



Create or capture complex screens with data-entry filters built in.

If all ScreenStar did was turn screens into code it would be a useful tool. Yet ScreenStar also permits a wide range of field types. Some of the choices include date, alphanumeric, telephone, yes/no, dollar, time and user-definable fields.

Other valuable data-entry filters are built in, such as required field, display only, and many others. All screen fields are generated with error-checking routines.

ScreenStar Not Only Captures Your Imagination, It Captures Screens.

The memory-resident capture program converts any screen into a ScreenStar file in seconds, including those generated by programs like Dan Bricklin's Demo Program.

ScreenStar Sets the Stage for Windows.

ScreenStar comes with a complete window generating library. You design the help screens and pop-up windows. Essential ScreenStar windowing functions tie them together in one smooth package.

Curtain Call.

They may not ask for your autograph, but they will want to know how you did those screens. Screenstar is more than a screen-painting program. It is a screen processor. No professional programming environment will be complete without this product.

We know you will enjoy using ScreenStar. However, should you give it less than rave reviews, return it within 30 days for a full refund.

★ Interactive screen painting and subsequent code generation.

★ Multiple screen design and scrolling.

★ TSR screen capture program, works with any program including Dan Bricklin's Demo Program.

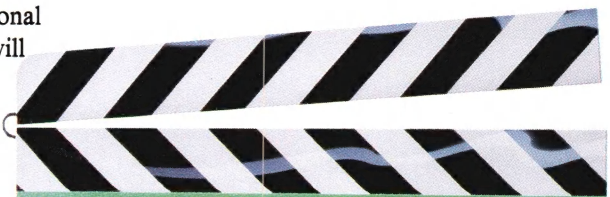
★ Complete window design including overlapping window functions.

★ Screens are compressed into data structures, and remain a permanent part of the program. No messy data files to look for.

Price - $99

W/Source add $99

Audition Our Product Today. Call:

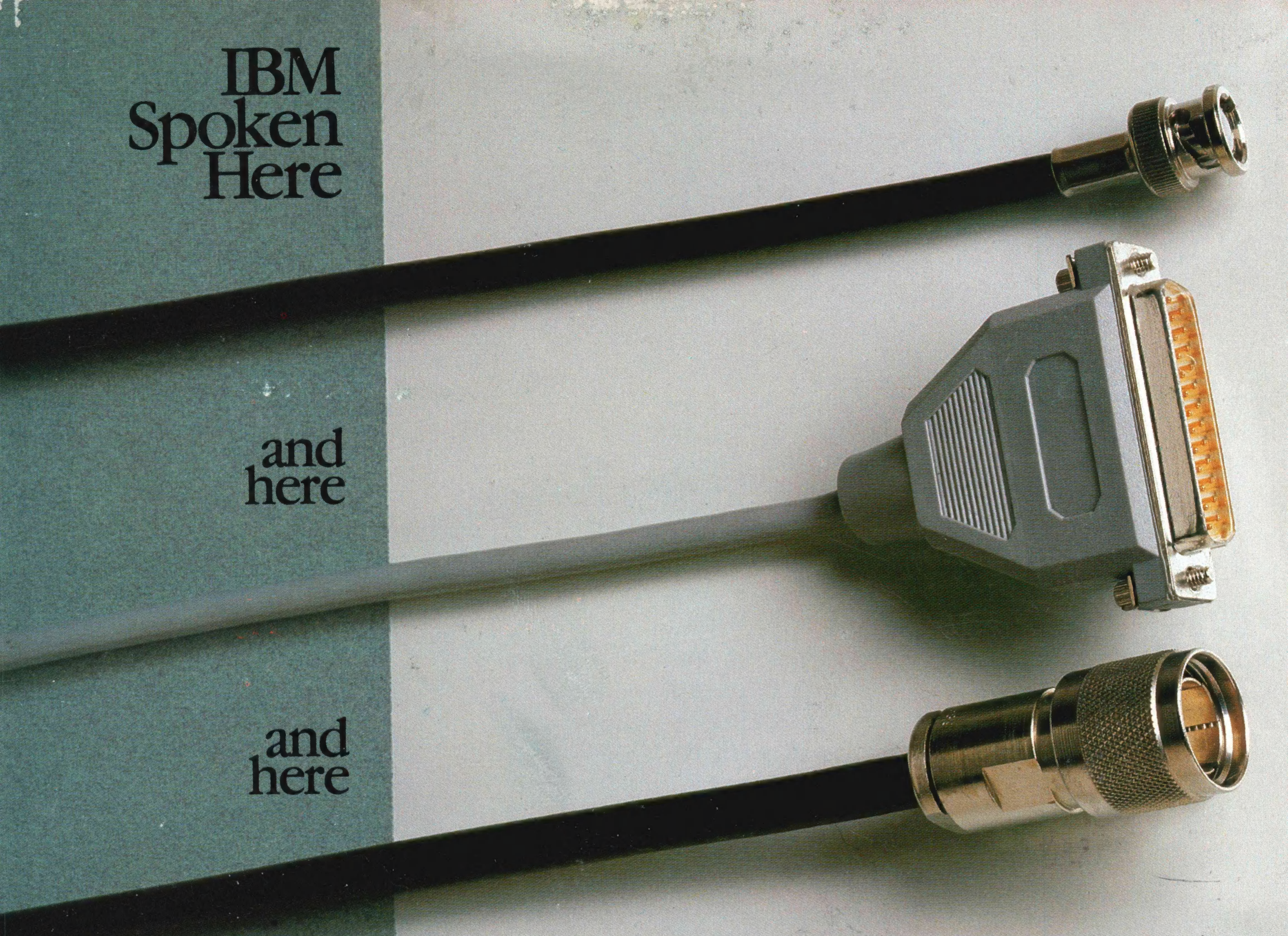


(201) 762-6965

Essential Software
South Orange Plaza
76 South Orange Ave., Suite 3
South Orange, NJ 07079

CIRCLE 139 ON READER SERVICE CARD

ScreenStar is a trademark of Essential Software Inc.
Dan Bricklin's Demo Program is a trademark of Software Garden Inc.



and
here

Whatever dialect of IBM you need to speak, CROSSTALK® Mk. 4 makes the connection.

Now, one program does the job that used to require several. CROSSTALK® Mk. 4 allows high-speed direct communications between PCs and minicomputers, or (with an IRMA™ board) between your PC and an IBM Mainframe, or (with Smart Alec™) between your PC and IBM System 3x's. If you like, CROSSTALK can support all of these sessions (and others) simultaneously, and display each session in its own window.

CROSSTALK Mk. 4 emulates all the terminals you're likely to find useful. That includes IBM 3101 (page and character modes), IBM 525x, IBM 529x, IBM 327x, as well as many popular async terminals like the DEC VT100 and VT220 series. CROSSTALK Mk. 4 includes the powerful CASL™ programming language, which allows you to automate communications applications quickly and easily.

So if you're used to thinking of CROSSTALK just to use with a modem, you're missing some important connections. Ask your dealer for details, or write:

dca® Digital Communications Associates, Inc.
1000 Holcomb Woods Parkway / Roswell, Georgia 30076
1-800-241-6393

CROSSTALK® COMMUNICATIONS

CROSSTALK and DCA are registered trademarks of Digital Communications Associates, Inc. IRMA, Smart Alec and CASL are trademarks of Digital Communications Associates, Inc. IBM is a registered trademark of International Business Machines Corp. DEC is a registered trademark of Digital Equipment Corp.

CIRCLE 171 ON READER SERVICE CARD